N° d'ordre : 2013telb0286

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En accréditation conjointe avec l'Ecole Doctorale Sicma

Etude et implémentation d'une architecture de décodage générique et flexible pour codes correcteurs d'erreurs avancés

---

Thèse de Doctorat

Mention : STIC

Présentée par **Jean Dion**

Département : Electronique

Laboratoire : Lab-STICC

Directeur de thèse : Michel Jézéquel

Soutenue le 5 novembre 2013

Jury :

M. Christophe Jégo, Professeur des Universités - IPB/ENSEIRB-MATMECA (Rapporteur)
M. Jean-Pierre Cances, Professeur des Universités, ENSIL (Rapporteur)
Mme Maryline Hélard, Professeur des Universités, INSA (Examinateur)
M. David Gnaedig, Ingénieur de recherche, TurboConcept (Examinateur)
Mme Marie-Hélène Hamon, Ingénieur de recherche, Orange Labs (Encadrante de thèse)
M. Pierre Pénard, Ingénieur de recherche, Orange Labs (Encadrant de thèse)
M. Matthieu Arzel, Maître de conférences, Télécom Bretagne (Encadrant de thèse)
M. Michel Jézéquel, Professeur Institut télécom, Télécom Bretagne (Directeur de thèse)

Remerciements

Après trois années de thèse au sein d'Orange Labs, je tiens à exprimer ma profonde reconnaissance à mes encadrants qui ont contribués à ces travaux avec l'apport de leurs visions et de leurs compétences. Je remercie particulièrement Marie-Hélène HAMON et Pierre PENARD pour leurs expériences et leurs sympathies, Matthieu ARZEL, qui m'a ouvert la voie du doctorat et dont le dynamisme m'a permis de dépasser mes limites, et le professeur Michel JEZEQUEL, mon directeur de thèse, pour sa disponibilité et son aide au long de cette expérience. Je tiens à vous exprimer ma sincère sympathie pour votre soutien et votre disponibilité tout au long de ce projet.

Je salue les membres de l'URD CREM qui m'ont accueilli pour l'élaboration de ce projet. Leurs points de vue sur les technologies dont ils sont experts ont permis d'enrichir cette thèse de nombreux cas d'usages. Je remercie particulièrement Jean-Christophe RAULT pour son accueil chaleureux et Marc LANOISELEE qui a élaboré la carte FPGA permettant les nombreuses intégrations de mes architectures.

J'exprime également ma sympathie aux membres du département ELEC de TELECOM Bretagne qui m'ont accueilli plusieurs semaines et m'ont transmis leurs expériences sur le codage de canal et le développement matériel.

Je remercie mon jury de thèse, les Professeurs Maryline HELARD, Christophe JEGO et Jean-Pierre CANCES, ainsi que David GNAEDIG.

Je salue Sanae, Akl, Sinda, Mohammed, Ali, Nahla, Lounes, Pierre, Redieteab, Hassan, Marc-Antoine, Tuan Anh et Senad.

Je remercie enfin mes parents, ma famille et mes amis pour leurs soutiens occasionnels ou réguliers.

Résumé

Mots clés : *turbocodes, codes QC-LDPC, décodage générique, décodage flexible, architecture FPGA, architecture ASIC, IEEE 802.11n, 3GPP LTE.*

Le codage de canal est une opération mathématique qui améliore la qualité des transmissions numériques en corrigeant les bits erronés en réception. Les contraintes des usages comme la qualité de réception, les débits d'utilisation, la latence de calcul, la surface ou encore la consommation électrique favorisent l'usage de différents codes dans la standardisation des protocoles de communication. La tendance industrielle est à la convergence des réseaux de communication pour des usages variés. Ce large choix de codage devient un handicap pour la conception de transmetteurs à bas coûts. Les réseaux médias favorisent des codes correcteurs d'erreurs avancés comme les turbocodes et les codes LDPC pour répondre aux contraintes de qualité de réception. Or ces procédés ont un coût de décodage important sur les récepteurs finaux. Une architecture adaptée à plusieurs types de codes capable d'évoluer en fonction d'une modification du protocole d'accès devient inévitable pour élaborer de nouveaux scénarios d'usages.
Ce mémoire présente le principe du codage de canal et la plupart des codes correcteurs d'erreurs avancés sélectionnés dans les standards de communication courants. Les caractéristiques communes des codes QC-LDPC et des turbocodes sont soulignées. Les principaux algorithmes ainsi que certaines architectures de décodage sont présentés. La complexité matérielle des principaux algorithmes de décodage est évaluée. Ils sont comparés pour un même code et à un niveau de correction équivalent pour les codes QC-LDPC. Une étude similaire est réalisée sur les turbocodes. Les algorithmes de décodage sont appliqués sur des codes de tailles et de rendements proches et dimensionnés pour atteindre une correction similaire afin de sélectionner un algorithme de décodage conjoint aux deux familles de code. Les codes QC-LDPC et les turbocodes se structurent à l'aide d'une représentation en treillis commune. La technique de fenêtrage couramment appliquée au décodage des turbocodes est étudiée pour le décodage d'un code QC-LDPC. Enfin, l'entrelacement des codes QC-LDPC est mis en évidence et reconsidéré en fonction des contraintes matérielles. Un cœur de décodage de treillis compatible avec les standards 3GPP LTE et IEEE 802.11n est proposé. Plusieurs structures de décodage sont ensuite introduites incorporant un ou plusieurs de ces cœurs. L'intégration sur cible FPGA est détaillée. Un scénario d'utilisation avec un contexte de décodage évoluant à chaque message reçu est proposé ce qui souligne l'impact de la reconfiguration sur les débits de décodage. La structure multistandard nécessite 4,2 % (respectivement 5,3 %) de ressources matérielles supplémentaires à une structure compatible avec le standard 3GPP LTE (resp. IEEE 802.11n) seul. La dégradation du débit maximal due à la reconfiguration entre le décodage des mots de code est d'au plus 1 %. Une architecture à plusieurs cœurs est également portée sur une cible ASIC de 65 nm. Cette architecture fonc-

tionne à une fréquence de 500 Mhz sur une surface de 2,1 mm^2 décodant les mots de code 3GPP LTE et IEEE 802.11n, et acceptant une reconfiguration dynamique entre deux mots de code consécutifs.

Abstract

Keywords : *turbo codes, QC-LDPC codes, generic decoding, flexible decoding, FPGA architecture, ASIC architecture, IEEE 802.11n, 3GPP LTE.*

Channel coding is a mathematical operation which improves the quality of the digital communication by correcting erroneous bits in reception. The usage constraints like a high quality reception, good throughputs, a small latency communication, a small silicon area or low power consumption promote the selection of a large variety of codes for standardized communication protocols. When industrial trend is to merge communication networks in order to answer a large panel of usage, the wide range of codes is a handicap to design a low cost transmitter. The media networks prefer advanced Forward Error Correction codes like turbo codes and LDPC codes to meet the constraints of signal quality. However, such an architecture has a huge hardware cost on the transmitter. A structure which fits to several kinds of codes and is able to adapt to an evolution of the medium protocol is not avoidable to design new usage scenarios.

This memory presents the principle of channel coding and several advanced forward error correction codes selected in the common standardized communication protocols. Common characteristics of QC-LDPC codes and turbo codes are underlined. The main algorithms and some decoding architectures are presented. The hardware complexity of the main decoding algorithms is estimated. They are compared for a given code with an equivalent correction capacity for QC-LDPC codes. A similar study is performed on the turbo codes. The decoding algorithms are then used to decode equivalent rate and length codes and laid out to achieve an equal correction capacity, in order to select a joint decoding algorithm fitting with the two families of code. The QC-LDPC codes and the turbo codes are structured thanks to a common trellis representation. The windowing technique commonly applied in turbo decoding is studied to decode a QC-LDPC code. Finally, the QC-LDPC interleaving are put in light and reconsidered in accordance with hardware constraints. A trellis decoding core compatible with 3GPP LTE and IEEE 802.11n standards is proposed. Several decoding structures are then introduced incorporating one or several cores. The integration on a FPGA target is detailed. A use-case scenario with a decoding context evolving every received message is proposed and highlights the impact of the reconfiguration on throughputs. The multi-standard structure requires 4.2 % (respectively 5.3 %) additive hardware resources in comparison with a single standard one compatible with 3GPP LTE (resp. IEEE 802.11n). The reconfiguration between two codewords from different standards reduces the throughputs by less than 1 %. A multi-cores architecture is also brought on a 65 nm ASIC target. This architecture operates at a frequency of 500 MHz on a 2.1 mm^2 silicon area, decoding codewords from 3GPP LTE and IEEE 802.11n standards, and accepting a dynamic reconfiguration between two consecutive codewords.

Table des matières

Table des figures

Liste des tableaux

Introduction

La plupart des réseaux de communication font intervenir des phénomènes physiques de déformation et d'atténuation de l'information émise et de perturbations externes liées aux conditions de transport ou aux équipements de communication. De ce fait, cette information est souvent corrompue à la réception. Bien que les mécanismes de synchronisation ou d'égalisation permettent de réduire les déformations du message physique, celui-ci comporte des erreurs qui peuvent s'avérer fatales pour la communication. Les mécanismes de correction d'erreurs référencés comme correction de canal ou mécanismes FEC (*Forward Error Correction*) permettent de réduire ces erreurs de communication aux moyens de procédés mathématiques. Différentes stratégies de codage de canal s'adaptent aux différents cas d'usage. Si certains codes simples permettent des usages adaptés pour les communications faiblement bruités, la plupart de ces codes ne conviennent pas aux exigences de performance des usages sur des réseaux bruités.

Les codes correcteurs d'erreurs les plus avancés nécessitent l'usage d'algorithmes de calcul itératifs. Ces processus permettent de réduire bien plus efficacement les erreurs de transmission. Différentes familles de codes correcteurs avancés s'adaptent à des cas d'usages variés. Il convient donc de favoriser le code le plus à même d'obtenir la meilleure correction possible tout en respectant les contraintes de latence et de débit des usages. Les critères matériels participent également à la sélection d'une stratégie de codage. Dans les années 1990, les premiers décodeurs de canal implantés sur carte représentaient plus de la moitié de l'espace attribué à l'architecture du récepteur. Leur bonne performance s'obtient alors au prix d'un coût matériel important. Les standards de communication récents comme les standards Wi-Fi (IEEE 802.11n, 802.11ac, IEEE 802.11ad) ou les standards de communication mobile (3GPP UMTS, 3GPP LTE) exploitent certains codes correcteurs d'erreurs avancés pour garantir des débits et une qualité de service.

Avec la numérisation de l'information, l'essor d'internet et des réseaux mobiles, les industriels envisagent de plus en plus de converger différents usages sur plusieurs réseaux existants. De multiples réseaux permettent d'améliorer la couverture Internet dans un logement tout en évitant des travaux de grandes ampleurs. L'adaptation des protocoles de communication pour une compatibilité entre les standards régissant le transfert de données sur réseaux locaux sans fils, sur courants porteurs en ligne ou encore sur câble Ethernet offre donc à l'usager une meilleure qualité de service à faibles coûts malgré la diversité des habitations. Dans un autre registre, les réseaux de communication locale peuvent être adaptés pour délester les réseaux de communication mobile qui sont de plus en plus sollicités par le grand public. Ces quelques exemples rendent inévitable l'adaptation des équipements d'émission et de réception pour une compatibilité avec plusieurs protocoles de communication standardisés. Or ceux-ci doivent également respecter des normes de dépenses énergétiques, des coûts de conception faibles et d'autres critères industriels.

Les multiples cas de compatibilité doivent donc être étudiés au plus bas niveau afin de mutualiser les éléments redondants. Les structures de décodage sont avant tout comparées par rapport à la réduction du nombre d'erreurs qu'elles permettent. Cette capacité de correction doit cependant être mise en perspective avec le débit de décodage atteint par la solution. La latence de traitement d'un message reçu est également un critère évalué en fonction des usages de la communication. Enfin, les aspects matériels comme la surface de silicium de la solution et sa consommation électrique influencent les coûts de conception et de fonctionnement du récepteur.

Dans ce mémoire, nous nous intéressons également à la capacité d'adaptation de l'architecture de décodage à différentes stratégies de codage. Ainsi, la généricité de la solution permet de vérifier l'adaptation d'une architecture de décodage à certaines d'entre elles. Cependant, cette adaptation ne doit pas dégrader la qualité de service globale de la structure. Ainsi, la flexibilité de la solution permet de quantifier l'adaptation de l'architecture à un changement de contexte de codage en termes de latence d'adaptation.

Ce travail de thèse se positionne sur le contexte de la convergence de stratégies de décodage. Pour répondre à cette problématique, nous nous sommes fixés comme objectif de mutualiser les ressources matérielles nécessaires à la convergence entre différents standards de communication tout en conservant des performances de décodage compétitives par rapport à d'autres solutions dédiées. Ce mémoire de thèse répond donc aux interrogations liées à la convergence entre des stratégies de décodage distinctes. Pour répondre à ces problématiques, ce mémoire s'articule sur quatre chapitres.

Dans un premier chapitre, nous définirons les principaux codes correcteurs d'erreurs avancés présents dans les standards de communication les plus courants. La problématique de la convergence de standard est également abordée d'un point de vue architectural.

Dans un second chapitre, les différents algorithmes de décodage dédiés à différentes représentations des codes sont abordés du point de vue de la propagation des informations et de leur complexité de calcul. Ceux-ci sont analysés en fonction des caractéristiques des codes. Des critères de comparaison d'algorithmes sont également proposés afin de sélectionner un algorithme de décodage conjoint à différentes stratégies de codage.

Dans un troisième chapitre, les critères de représentation en treillis des codes correcteurs d'erreurs avancés sont présentés afin de mettre en évidence les différents niveaux de parallélisme de calcul et de bénéficier d'une mutualisation des ressources élevée pour répondre à un cas d'usage prédéfini. Cette étape met donc en place les éléments nécessaires à la réalisation d'une architecture de décodage conjointe.

En dernier lieu, la réalisation matérielle d'un cas d'usage défini est présentée. Les principaux critères de performances de décodage et de performance matérielles sont fournis et comparés à d'autres solutions existantes.

Enfin, nous résumerons l'ensemble des contraintes fixées pour cette étude et les résultats obtenus. Les perspectives de ce travail ainsi que les pistes de recherches futures seront également détaillées pour répondre à d'autres besoins.

Chapitre 1

Vers un récepteur multistandard

L'utilisation de terminaux multistandards devient de plus en plus nécessaire pour répondre à des cas d'usage diversifiés. Or, le codage de canal définit dans les différents standards fait appel à des stratégies de protection de données spécifiques aux types de canal de transmission. La structure de décodage associée à certaines d'entre elles requiert la mobilisation de nombreuses ressources matérielles. Ainsi, la sélection de mécanismes dédiés défavorise la conception matérielle des terminaux multistandards. Une architecture de décodage générique s'avère être un atout concurrentiel nécessaire pour la conception de terminaux multistandards en termes de coûts de conception et de consommation électrique.

Ce premier chapitre aborde la diversité des solutions de codage de canal en fonction des différents protocoles de communication standardisés ainsi qu'en fonction de scénarios d'usage plausibles. Pour répondre à la problématique du décodage multistandard, il convient de caractériser le codage de canal dans sa globalité. Les codes correcteurs avancés permettent une protection des données très efficace. Ils font appel à des algorithmes de décodage distincts dont la réalisation matérielle est coûteuse en ressources de calcul. Ce chapitre synthétise les principaux codes correcteurs d'erreurs avancés ainsi que leurs stratégies de décodage. Une discussion sur les architectures de décodage existantes et sur les cibles matérielles est reprise afin de souligner les enjeux d'une architecture de décodage multistandard dont le contexte puisse évoluer sans impacter la latence de décodage.

1.1 Introduction

La correction de canal est devenue une étape essentielle pour protéger les données émises sur les canaux de transmission bruités. Les stratégies de codage s'adaptent cependant en fonction des contextes de communication et du type de données transmises. Dans cette partie, un état de l'art montre la multiplicité des stratégies de codage de canal dans les différents standards de communication et établit des pistes de convergence pour lesquelles des récepteurs multistandards doivent être considérés.

1.1.1 Codes et standards

Un système de communication numérique est conçu pour modéliser le transfert d'informations numériques sur des canaux de transmissions bruités. Concrètement, lorsqu'une information est transmise sur un canal, elle est soumise à de nombreuses perturbations physiques qui entraînent une déformation à la réception du signal transmis. Des erreurs résultent de ces perturbations, et le message reçu n'est pas considéré comme fiable. L'enjeu du codage de canal est de pouvoir détecter puis de corriger ces erreurs. Pour ce faire, Shannon [1] a établi la théorie mathématique des communications. Dans son ouvrage, Shannon développe les règles du codage de canal, et prouve l'existence d'une limite théorique qu'un code correcteur d'erreur puisse atteindre.

Depuis l'établissement de la théorie des communications, de nombreux codes ont été inventés puis inclus dans les standards de communication.

Les codes de Hamming [2], de Golay [3] ou les codes BCH [4] sont des codes de très petite dimension, de l'ordre de la dizaine de bits. Le code de Golay a permis aux sondes Voyager de transmettre les premières images de Jupiter. Le code de Hamming est actuellement employé dans le standard IEEE 802.15.1 [5]. Les codes de Reed-Solomon [6] sont des codes en blocs de tailles variables dont le calcul s'effectue dans un corps de Galois. Ils sont actuellement sélectionnés pour le stockage de données sur CD et DVD.

Les besoins de communication ont fortement augmenté depuis les années 1980, et les nouveaux standards de communication intègrent les dernières avancées de la recherche sur le codage correcteur d'erreurs. Ainsi, les années 1980 standardisent des solutions de codage avec concaténation d'un code en bloc externe (en général un code de Reed-Solomon) et un code interne de type convolutif. Ce double codage permet d'obtenir de bonnes performances de décodage. On les retrouve notamment sur des standards de vidéo numérique de diffusion du type DVB-C [7], DVB-T [8] et DVB-S [9].

Les performances de décodage se rapprochent graduellement de la limite de Shannon, jusqu'à $3,5$ dB. Les turbocodes apparaissent dès 1992 [10], affichant des performances approchant cette limite à $0,5$ dB, pour un code constituant binaire à 16 états, et après 18 itérations de décodage. Cette découverte relance la recherche vers les codes LDPC de R.G. Gallager [11] qui furent standardisés pour la première fois dans le standard de diffusion par satellite DVB-S2 [12].

Les deux technologies de codage de canal sont souvent opposées dans les comités de normalisation. Les codes LDPC sont étudiés pour rechercher les meilleures performances, en travaillant notamment sur la structure des codes, et sur les architectures de décodage. Les codes LDPC Quasi-Cycliques [13] deviennent les codes LDPC les plus répandus. La TABLE 1.1 liste certains codes LDPC standardisés. Du coté des turbocodes, l'évolution de la technologie matérielle a permis de privilégier des codes convolutifs double-binaires comme codes constituants. Ces codes sont inclus dans de nombreux standards actuels référencés dans la TABLE 1.2.

Les codes LDPC et les turbocodes sont les deux familles de **codes correcteurs d'erreurs avancés** régulièrement sélectionnés par les comités de standardisation, et en particulier pour des standards de communication sur des canaux de transmissions fortement bruités comme les réseaux radio (RAN) ou les réseaux sur courants porteurs en ligne (PLC).

Standards	**Famille de code**	**Nombre de matrices**	**Facteur d'expansion** z	**Taille max. des mots de code**	**Nombre max. de nœuds**
Standards IEEE					
802.11n [14]	QC	12	27, 54, 81	1944	7128
802.11ac [15]	QC	12	27, 54, 81	1944	7128
802.11ad [16]	QC	1	42	672	2352
802.16m [17]	QC	6	24, 28, ..., 96	2304	8448
802.15.3c [18]	QC	4	21	672	2562
Standards ITU-T					
G.Hn [19]	QC	7	$14, ..., 360$	8640	22140

TABLE 1.1: Caractéristiques des codes QC-LDPC dans les standards

1.1.2 Scénarios multistandards

La section 1.1.1 donne un aperçu des choix de codes correcteurs d'erreurs dans de nombreux standards de communication. Les codes correcteurs d'erreurs avancés font intervenir des algorithmes de décodage itératifs dont l'intégration sur puces électroniques est considéré comme complexe. Pourtant, la dernière décennie montre l'émergence de solutions de communication permettant de multiples usages impliquant des réseaux de communication différents.

Le domaine des réseaux domestiques multimédia montre l'intérêt de multiplier les canaux de communications devant la diversité des infrastructures réseaux existantes. La communication Ethernet est associée avec la communication sur réseau local sans-fil (WLAN) et la communication sur courants porteurs en ligne (PLC).

Standard	Valeur m	Nombre d'état	Fermeture de treillis	Taille des treillis	Taille max. des treillis
Standard propriétaire					
CCSDS [20]	1	16	Retour à zéro	$K+4$	16388
Standards mobiles 3GPP et 3GPP2					
UMTS HSDPA [21]	1	8	Retour à zéro	$K+3$	5117
LTE LTE-A [22]	1	8	Retour à zéro	$K+3$	6117
CDMA2000 [23]	1	8	Retour à zéro	$K+3$	5147
Standards IEEE					
802.16m [17]	2	8	État circulaire	$K/2$	765
Standards DVB					
DVB-RCS [24]	2	8	État circulaire	$K/2$	752
DVB-RCT [25]	2	8	État circulaire	$K/2$	324
DVB-SH [26]	1	8	Retour à zéro	$K+3$	12285
DVB-RCS2 [27]	2	16	État circulaire	$K/2$	1599
Standards PLC					
HomePlug AV [28]	2	8	État circulaire	$K/2$	2080
Homeplug AV2 [29]	2	8	État circulaire	$K/2$	2080

TABLE 1.2: Caractéristiques des turbocodes convolutifs dans les standards

Le groupe de standardisation IEEE 802.11 définit les technologies de transmission radio WLAN, plus connues sous l'appellation Wi-Fi. Ce groupe de travail génère une profusion de standards dont on peut citer le IEEE 802.11n [14], son évolution IEEE 802.11ac [15] et son adaptation sur la bande des 60 GHz IEEE 802.11ad [16]. Les codes LDPC standardisés dans la version IEEE 802.11ac sont rétro-compatibles avec le standard IEEE 802.11n alors que le standard IEEE 802.11ad privilégie d'autres matrices.

Le groupe de standardisation IEEE 802.3 [30] définit la technologie Ethernet. Ce groupe privilégie une protection des données par des codes de Reed-Solomon.

La standardisation sur réseaux PLC est étudiée par deux grands groupes de travail. Le comité de standardisation ITU-T travaille sur la spécification G.Hn sur l'utilisation des réseaux domestiques regroupant la communication sur courants porteurs en ligne, sur câbles téléphoniques et sur câbles coaxiaux. La recommandation G.9960 [19] regroupe les contraintes liées à la couche physique. Ce standard favorise une protection à base de codes QC-LDPC. Le groupe Homeplug Alliance travaille sur les spécifications Homeplug dont HomePlug AV [28] et HomePlug AV2 [29]. Ces standards privilégient l'usage d'un turbocode double-binaire à 8 états. Le groupe de standardisation IEEE reprend les travaux de ce groupe dans son propre standard IEEE 1901 [31].

La multiplication des usages domestiques rend l'interconnexion de ces réseaux de plus en plus envisagée. Le groupe IEEE 1905.1 [32] a travaillé sur une couche d'abstraction située entre la couche MAC et la couche réseau afin d'uniformiser l'usage des accès gérés par les groupes IEEE 802.3, IEEE 802.11, IEEE 1901 et MoCA (*Multimedia over Coax Alliance*). Ce projet facilite la convergence de standards sur les réseaux domestiques et l'émergence de terminaux multi technologies. Les couches physiques de ces standards n'ont cependant pas été modifiées pour appliquer une protection de données unifiée. Une architecture apte à décoder un turbocode double-binaire à 8 états (spécifications HomePlug) et les codes LDPC (spécifications IEEE 802.11) présente donc un cas d'usage adapté à ce projet de convergence multistandard.

La communication mobile présente également des cas d'usages multistandards. Le groupe 3GPP travaille à améliorer les réseaux mobiles sur les fréquences licenciées. En faveur de son expérience des standards avec UMTS puis HSPA [21], ce groupe a proposé les standards 3GPP LTE puis 3GPP LTE-A [22] pour répondre au projet de quatrième génération des réseaux mobiles lancée par l'ITU-T. Ces standards privilégient une correction d'erreur par turbocode binaire. Malgré les gains en capacité également apportés par ces solutions, la demande croissante de débits par les utilisateurs augmente graduellement la charge de ces réseaux. De ce fait, les opérateurs de téléphonie étudient les réseaux de communication sans-fils locaux dominés par le groupe IEEE 802.11 afin de désengorger les réseaux mobiles. Les appareils de communication compatibles avec ces deux technologies sont présents sur le marché à l'instar des smartphones. Ce projet d'étude entre dans la réduction du coût de fabrication de ce type d'appareils multi usage en intégrant un décodeur de canal apte à modifier son mode de fonctionnement à la volée.

La convergence d'usages de réseaux présente un intérêt grandissant. Le coût matériel et financier des adaptations aux différents standards pousse les constructeurs à mutualiser les ressources nécessaires. Le décodage de canal présente dans ce cas un coût matériel important. L'accès à une technologie de décodage générique et flexible est donc un atout industriel qu'il convient d'étudier afin d'en déterminer la complexité et les contraintes d'architecture.

1.2 Caractérisation du codage de canal

Les codes correcteurs sont caractérisés en fonctions de critères communs aux différentes stratégies de codage. Cette partie aborde les principes généraux de la communication sur un canal numérique ainsi que les premiers éléments qui caractérisent la correction de données. Les éléments rappelés dans cette section sont communs aux stratégies de codage de canal présentées dans la section 1.1.

1.2.1 Chaîne de communication numérique

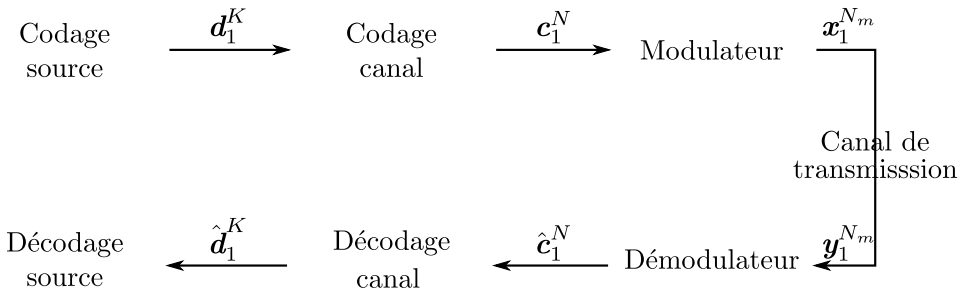


FIGURE 1.1: Modélisation d'une chaîne de transmission numérique

On définit un système de communication numérique comme l'ensemble des fonctionnalités appliquées au transfert de messages entre une source et un récepteur. Une vision simplifiée de ce système est illustrée sur la FIGURE 1.1. Dans un tel système, on suppose l'existence d'un émetteur souhaitant envoyer un message numérique constitué de K éléments binaires. Ce message est alors noté $\boldsymbol{d}_1^K = \{d_1, ..., d_K\}$. La chaîne de communication décrit le processus d'acheminement de ce message à travers un canal.

Dans un premier temps, le message émis $\boldsymbol{d}_1^K$ subit une opération de codage de canal. Le codage de canal vise à protéger le message de diverses déformations du message lors du transfert de celui-ci. Le message source est transformé en un ensemble de N éléments binaires. Ce message est noté $\boldsymbol{c}_1^N = \{c_1, ..., c_N\}$. Cette transformation est discutée dans la suite de ce chapitre.

Le message $\boldsymbol{c}_1^N$ est ensuite adapté au canal de transmission. Les N éléments binaires de $\boldsymbol{c}_1^N$ sont transformés en N_m symboles de modulation numérique. Ce message modulé est noté $\boldsymbol{x}_1^{N_m} = \{x_1, ..., x_{N_m}\}$.

Le message modulé $\boldsymbol{x}_1^{N_m}$ subit de nombreuses perturbations au cours de la transmission. La modélisation du **bruit** sur le canal est discutée dans la partie 1.2.3. La séquence reçue contient N_m éléments. Elle est notée $\boldsymbol{y}_1^{N_m} = \{y_1, ..., y_{N_m}\}$.

Le démodulateur transforme ces éléments en une série de décisions binaires (ou **décisions dures**) de N bits notée $\hat{\boldsymbol{c}}_1^N = \{\hat{c}_1, ..., \hat{c}_K\}$ ou en une série de N éléments réels (ou **décisions souples**). Ces informations sont utilisées par un module de décodage de canal qui prend une décision sur les K éléments binaires émis. Le résultat est noté $\hat{\boldsymbol{d}}_1^K = \left\{\hat{d}_1, ..., \hat{d}_K\right\}$. Les différentes transformations opérées par la chaîne de transmission sont discutées tout au long de cette section.

1.2.2 Modulations numériques du message

Cette section s'intéresse aux modulations numériques les plus courantes. La modulation numérique correspond à l'adaptation des signaux du domaine numérique au domaine analogique. Ce phénomène se traduit par la modulation d'une onde sinusoïdale en amplitude ou en phase. Cette transformation est représentée par une information complexe. Un **diagramme de constellation** aussi nommé diagramme IQ représente dans ce cas le codage des éléments binaires en éléments complexes. On distingue plusieurs types de modulations.

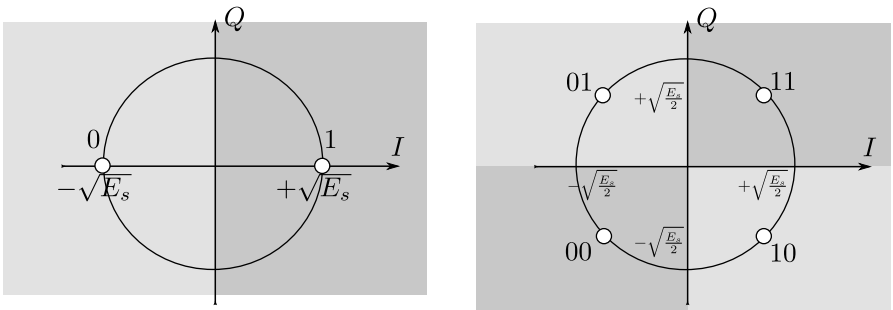
Les modulations par saut de phase (PSK) représentent les n_m éléments de modulation sur un cercle de même amplitude et répartis suivant les racines $n_m{}^{imes}$ de l'unité. On distingue en particulier la modulation BPSK (*Binary Phase-Shift Keying*) définie par (1.1), QPSK (1.2) (*Quadratic Phase-Shift Keying*), 8-PSK, ... Pour ces types de modulation, l'amplitude de chaque élément est constante et donnée en fonction de l'énergie nécessaire à l'émission d'un symbole numérique E_s. La FIGURE 1.2 représente différents diagrammes de constellation associés à ces modulations.

$$x_n = \sqrt{E_s} \times (2.c_n - 1) \tag{1.1}$$

$$x_n = \sqrt{E_s} \times (2.c_{2n} - 1) + j.\sqrt{E_s} \times (2.c_{2n+1} - 1) \tag{1.2}$$

La propagation du canal entraîne une déformation de la constellation entre l'émission et la réception. Le démodulateur établit une première décision à la réception en fonction du diagramme IQ notée $\hat{c}_n$. Pour ce faire, le diagramme de constellation est séparé en zones de décisions. La FIGURE 1.2 montre les secteurs de décision suivant ce principe pour les modulations BPSK et QPSK. Cette décision au niveau binaire est appelée **décision dure**.

La décision dure entraîne une forte perte d'information des éléments reçus. En effet, la notion de distance entre un élément reçu y_n et sa décision dure $\hat{c}_n$ est perdue. On peut donc améliorer cette décision en définissant une relation de distance entre les éléments. On définit le **Log-Rapport de Vraisemblance** noté $\mathrm{L}^c(c_k)$ comme le rapport de probabilités entre les différentes décisions sur les éléments binaires



(a) (b)

FIGURE 1.2: Diagramme de constellation pour une modulation BPSK (a) et QPSK (b)

en fonction des informations du canal. Ce Log-Rapport de Vraisemblance est défini comme (1.3) pour $c_k \in \{0, 1\}$ et (1.4) sinon.

$$\mathrm{L}^{c}\left(c_k\right) \triangleq \log\left(\frac{\Pr\left\{c_k = 1 | y_k\right\}}{\Pr\left\{c_k = 0 | y_k\right\}}\right) \tag{1.3}$$

$$\mathrm{L}^{c,i}\left(c_k\right) \triangleq \log\left(\frac{\Pr\left\{c_k = i | y_k\right\}}{\Pr\left\{c_k = 0 | y_k\right\}}\right) \tag{1.4}$$

Dans la suite de ce mémoire, la modulation BPSK est privilégiée comme modulation de référence.

1.2.3 Canal de transmission

Le canal de transmission représente les différentes perturbations engendrées lors de la transmission du signal entre une source et le récepteur. Cette représentation caractérise toutes les déformations physiques de la transmission.

Il existe de nombreuses représentations des perturbations du canal. Le canal à bruit blanc additif gaussien (AWGN) est le canal de référence pour étudier les codes correcteurs d'erreurs et leurs algorithmes de décodage. Le canal AWGN suppose qu'un élément de bruit b_n est ajouté à l'élément émis x_n. L'expression (1.5) résume le phénomène de transmission sur canal AWGN.

$$y_n = x_n + b_n \tag{1.5}$$

L'élément de bruit b_n suit une loi Normale $\mathcal{N}$ complexe de moyenne μ nulle et de variance σ^2. La densité de probabilité de cette loi est établie par la formule (1.6).

$$f\left(y_n\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{-\frac{(y_n-\mu)^2}{2\sigma^2}} \tag{1.6}$$

La puissance de bruit dépend de la densité du spectre de bruit N_0. De ce fait, $\sigma^2 = N_0/2$. Le canal AWGN est sans mémoire. Chaque élément de la constellation numérique reçu est décorrélé de ces voisins.

Le modèle de transmission par canal AWGN prend en compte le schéma de modulation mis en place et défini dans la section 1.2.2. Pour une transmission sur un canal AWGN de moyenne μ nulle et de variance σ^2 avec modulation BPSK, le Log-Rapport de vraisemblance $\mathrm{L}^c(c_k)$ correspond à l'expression (1.7).

$$\mathrm{L}^c(c_n) = \frac{2}{\sigma^2} y_n \tag{1.7}$$

L'utilisation d'un canal AWGN est supposée tout au long de ce manuscrit. Ce choix correspond aux conditions d'expérimentation du décodage de canal pour la plupart des codes standardisés.

1.2.4 Codage de canal

La correction de canal implique d'ajouter des bits supplémentaires (appelés bits de parité ou données de redondance) par rapport aux bits d'information (alors appelés bits systématiques) du codage source. Le message d'information $\boldsymbol{d}_1^K$ de K bits est transformé en un message codé $\boldsymbol{c}_1^N$ de N bits. Le codage de canal consiste donc à ajouter $M = N - K$ bits de redondance. On définit le **rendement de codage** R par la relation (1.8).

$$R \triangleq \frac{K}{N} \tag{1.8}$$

Le codage de canal pour un code $\mathcal{C}$ de longueur finie se caractérise par une fonction mathématique $\mathbf{c}$. Lorsque cette fonction est une application linéaire, les codes correcteurs associés sont dits **linéaires**. Pour ce type de code, il existe une matrice G qui permet de définir la fonction de codage $\mathbf{c}$. Cette matrice est nommée **Matrice Génératrice** du code $\mathcal{C}$. La fonction de codage est définie par la relation (1.9). De plus, l'espace des images des codes $\mathcal{C}$ est un sous-espace vectoriel de l'espace $\mathcal{A}_2^N$.

$$\begin{array}{rccl} \mathbf{c}: & \mathcal{A}_2^K & \rightarrow & \mathcal{C} \subset \mathcal{A}_2^N \\ & c & \mapsto & x = c.G \end{array} \tag{1.9}$$

La distance de Hamming d_H est définie dans cet espace par la relation (1.10).
$\forall \boldsymbol{c}_{\mathbf{a}1}^N, \boldsymbol{c}_{\mathbf{b}1}^N \in \mathcal{C}$

$$d_H\left(\boldsymbol{c}_{\mathbf{a}1}^N, \boldsymbol{c}_{\mathbf{b}1}^N\right) \triangleq \sum_{n=1}^{N} |c_{\mathbf{a}n} - c_{\mathbf{b}n}| \tag{1.10}$$

La distance de Hamming permet d'établir une relation de distance entre deux éléments de $\mathcal{C}$. Ainsi, la **distance minimale** (1.11) d'un code représente la plus petite distance de Hamming entre les mots d'un code. La linéarité des codes permet

d'établir que la différence entre deux mots de codes est un mot de code. De ce fait, la distance de Hamming se résume à la seconde expression de l'équation (1.11).

$$\begin{aligned} d_{min} &\triangleq \min_{\boldsymbol{c}_{\mathbf{a}_1}^N, \boldsymbol{c}_{\mathbf{b}_1}^N \in \mathcal{C}} \sum_{n=1}^{N} |c_{\mathbf{a}n} - c_{\mathbf{b}n}| \\ &= \min_{\boldsymbol{c}_1^N \in \mathcal{C}} \sum_{n=1}^{N} |c_n| \end{aligned} \tag{1.11}$$

Les codes en blocs et les codes convolutifs usuels sont des codes linéaires. Ces premiers paramètres permettent de caractériser la plupart des codes correcteurs à travers le triplet (N, K, d_{min}).

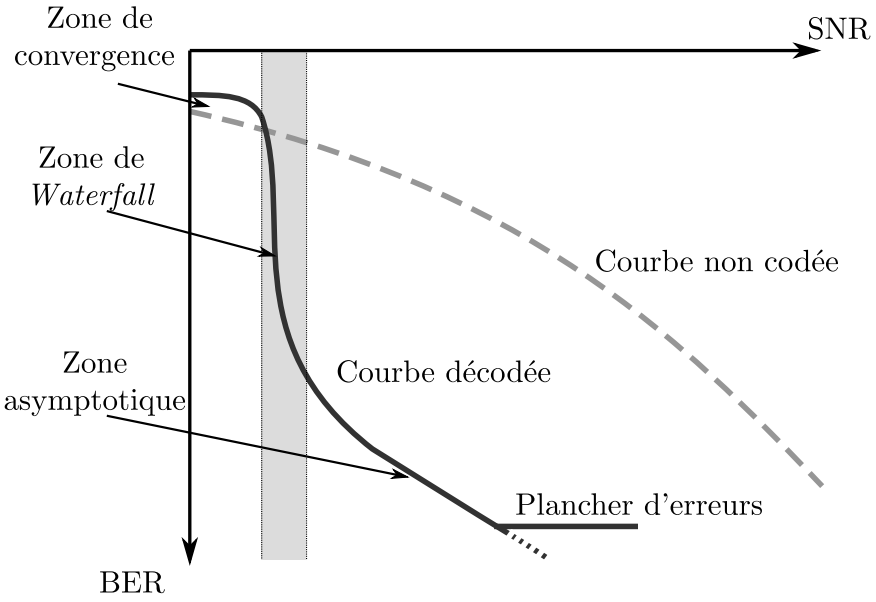


FIGURE 1.3: Courbe de performance de décodage caractéristique

Le codage de canal est une opération qui permet de réduire les erreurs de propagation du canal. Il reste encore à établir les métriques permettant de quantifier ces erreurs de propagation.

Le rapport signal à bruit (SNR) définit la différence entre la puissance du signal émis et le bruit du canal de propagation. Celle-ci est exprimée en décibel (dB). La puissance du signal découle de l'énergie nécessaire à l'émission d'un symbole de modulation E_s. La puissance de bruit dépend de la densité du spectre de bruit N_0. Ainsi, le SNR se définit suivant la relation (1.12).

$$\begin{aligned} SNR &= \frac{E_s}{N_0} \\ SNR_{dB} &= -20.\log_{10}\left(\frac{E_s}{N_0}\right) \end{aligned} \tag{1.12}$$

Afin de comparer les performances en sortie de décodage, on s'intéresse non pas au rapport SNR, mais au rapport E_b/N_0 qui rend compte de l'énergie E_b nécessaire

à l'émission d'un élément binaire de la source d_k par rapport au bruit du canal. Dans ce cas, l'énergie binaire vérifie la relation (1.13).

$$E_b = \frac{E_s \times N}{n_m \times K} \tag{1.13}$$

En reprenant un schéma de transmission sur un canal AWGN avec une modulation BPSK, la variance de bruit est calculée en fonction de ces paramètres, dont l'expression devient (1.14).

$$\sigma^2 = \frac{E_s \times N}{n_m \times K} \times 10^{-\frac{SNR}{20}} \tag{1.14}$$

La performance d'un code est établie en fonction de deux métriques. Le taux d'erreur binaire (BER) est défini en fonction du nombre de bits erronés en sortie du décodeur par rapport au nombre de bits émis total. Le taux d'erreur trame (FER) est défini en fonction du nombre de trames de K bits erronées en sortie du décodeur par rapport au nombre de trames émises. Ces quantités suivent les relations (1.15-1.16).

$$BER = \frac{\text{Nombre d'erreurs binaires}}{\text{Nombre de bits émises}} \tag{1.15}$$

$$FER = \frac{\text{Nombre d'erreurs trames}}{\text{Nombre de trames émises}} \tag{1.16}$$

Les performances de décodage sont représentées par des courbes de BER ou de FER par rapport au bruit SNR ou E_b/N_0 comme représenté sur la FIGURE 1.3. Sur cette figure, deux courbes représentent le taux d'erreur avant et après codage. Le gain de codage représente le gain en décibel pour atteindre un taux d'erreur fixé. Sur les courbes de décodage, trois comportements sont visibles. À faible SNR, le décodage n'entraîne pas de gain de décodage. Bien souvent, le taux d'erreur est plus important que dans le cas sans codage. À partir d'une valeur SNR, la courbe de taux d'erreur effectue une chute. Cette zone est appelée Zone de chute d'eau ou zone de *Waterfall*. Le seuil SNR est à comparer avec la limite théorique calculée par Shannon. Cependant, la courbe de décodage conserve un comportement asymptotique à partir d'un certain seuil BER. Les architectures matérielles de décodage peuvent également présenter un plancher d'erreurs qui correspond alors à un nombre d'erreurs dû à des problématiques d'implantation.

1.2.5 Caractéristiques des codes correcteurs

Les codes correcteurs d'erreur sont regroupés suivant leurs caractéristiques et leurs propriétés. La première grande caractéristique concerne leur propriété de linéarité. Cette propriété est vérifiée pour tout code étudié dans ce mémoire. Une autre propriété concerne le caractère systématique d'un code.

Définition 1. *Un code est dit* ***systématique*** *si le message à encoder* $\mathbf{d}_1^K$ *est contenu dans le mot encodé* $\mathbf{c}_1^N$.

Pour des codes correcteurs systématiques, le mot de code est donc divisé en K bits systématiques et $M = N - K$ bits de redondance.

L'espace de calcul entre également en jeu dans la caractérisation des codes correcteurs. On définit un **alphabet** $\mathcal{A}$ qui représente l'ensemble des valeurs que peut prendre un élément de calcul de base pour un code $\mathcal{C}$. Les codes **binaires** utilisent l'alphabet binaire $\mathbb{F}_2 = \{0, 1\}$. Certains codes utilisent les propriétés des corps de Galois. Dans ce cas, les éléments binaires sont regroupés en groupes de puissance de deux. Ces codes sont alors définis comme des codes **non-binaires**, et leur alphabet correspond au corps de Galois noté $\mathbb{F}_{2^q}$ ou $GF(2^q)$. Certains codes regroupent les éléments binaires en couples. On parle de codes **double-binaires**. Ce groupe de codes utilise la représentation $\mathcal{A} = \{00, 01, 10, 11\}$.

La flexibilité des codes correcteurs est également un facteur de distinction. Les codes en blocs sont des codes de longueurs fixe. De ce fait, un code $\mathcal{C}$ se caractérise par un couple (K, N) prédéfini. A l'inverse, les codes convolutifs sont des codes basés sur des longueurs non établies et donc potentiellement infinies. Dans le cadre des codes correcteurs d'erreurs avancés, les codes LDPC font partie de la famille des codes en blocs, tandis que les turbocodes convolutifs réutilisent le principe des codes convolutifs.

1.3 Familles de codes correcteurs d'erreurs avancés

Les protocoles de communication standardisés font intervenir une grande variété de stratégie de codage de canal. Les codes correcteurs d'erreurs avancés permettent d'obtenir de bonnes performances de décodage au prix d'une complexité accrue. Cette section détaille les caractéristiques des deux familles de codes couramment employées que forment les codes LDPC et les turbocodes au moyen des critères énoncés dans la section 1.2.

1.3.1 Les Codes LDPC

1.3.1.1 Principes généraux

Les codes LDPC ont été inventés par R.G.Gallager [11] en 1963 dans le but d'augmenter la distance minimale d_{min} des codes en blocs et ainsi de se rapprocher des limites de performances théoriques fixées par Shannon [1] en 1948. Ils ont été redécouvert par Mackay dans les années 1990 [33], qui voient naître dès lors de nombreux codes LDPC dont les caractéristiques varient suivant les problématiques envisagées. MacKay et Neal [34] définissent des codes MN largement inspirées des codes LDPC, et Wilberg [35] établit une représentation commune sur un graphe de Tanner. Richardson [36] étudie la répartition des codes LDPC ainsi que leur régularité afin d'en améliorer leurs performances de décodage. En 2001, Ten Brink étudie la métrique extrinsèque aux travers de l'EXIT Chart [37]. [13] propose de structurer les codes LDPC pour limiter la complexité d'encodage et de décodage. Il instaure la famille des codes LDPC Quasi-Cycliques notés QC-LDPC.

Les codes LDPC sont caractérisés par la définition 2.

Définition 2. *Un code LDPC (Low-Density Parity-Check) est un code en bloc dont la matrice de parité H est considérée comme peu dense. Sa matrice de parité H contient un faible nombre d'éléments non-nuls.*

La matrice de parité H, de taille $M \times N$ et définie par ces coefficients $(h_{m,n})$ est dimensionnée en fonction de la caractéristique des codes. La matrice possède exactement N colonnes et $(N-K)$ lignes. Tout mot de code vérifie la relation (1.17).

$\forall \boldsymbol{c}_1^N \in \mathcal{C}$,

$$H.\boldsymbol{c}_1^{N^T} = 0 \tag{1.17}$$

En reprenant les notations de la chaîne de communication introduite dans la partie 1.2.1, la relation (1.17) équivaut à un ensemble de $(N-K)$ équations de parité. Chacune de ces équations vérifie la relation (1.18).

$\forall \boldsymbol{c}_1^N \in \mathcal{C}$,

$$\sum_{n=0}^{N-1} h_{m,n}.c_n = 0 \tag{1.18}$$

Afin de mieux organiser cette matrice, les éléments non-nuls de H sont repérés par leurs indices de ligne m et colonne n. Dans ce cas, l'ensemble d'indices des variables impliquées dans une équation m, repéré par l'ensemble $\mathcal{N}_m$ vérifie donc la relation (1.19).

$\forall m \in [\![0; M[\![$,

$$\mathcal{N}_m \triangleq \{n \in [\![0; N[\![, h_{m,n} \neq 0\} \tag{1.19}$$

De même, on définit l'ensemble des indices des équations où l'élément codé c_n est impliqué par l'ensemble $\mathcal{M}_n$. Cet ensemble vérifie la relation (1.20).

$\forall n \in [\![0; N[\![$,

$$\mathcal{M}_n \triangleq \{m \in [\![0; M[\![, h_{m,n} \neq 0\} \tag{1.20}$$

[36] étudie la répartition des éléments non-nuls d'une matrice de parité. Il réutilise les notions de degrés de variable et de degrés de parité associés à une matrice définis par [38] et en étudie la répartition afin d'améliorer les performances BER de ces codes.

Définition 3. *Le degré de variable associé à une équation e_m est défini comme le nombre de variables impliquées dans cette équation. Le degré de variable est noté d_v^m.*
Le degré de parité associé à un élément c_n est défini comme le nombre d'équations de parité dans laquelle l'élément c_n est impliqué. Le degré de parité est noté d_c^n.

Le degré de variable d_v^m correspond au cardinal de l'ensemble $\mathcal{N}_m$. De même, le degré de parité d_c^n correspond au cardinal de l'ensemble $\mathcal{M}_n$.

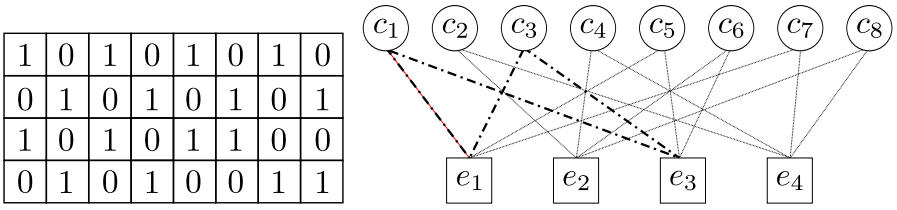
[36] propose de distinguer les codes LDPC en fonction de ses degrés de variables et de parités. Une matrice de parité est dite **régulière** si ses degrés de variable d_v^m

et ses degrés de parité d_c^n sont constants. Dans le cas contraire, la matrice est dite **irrégulière**. [36] montre également que les matrices irrégulières offrent de meilleurs performances par rapport aux matrices régulières.

1.3.1.2 Représentation sur Graphe de Tanner

Les codes linéaires dont on connaît une matrice de parité se représentent sous une forme graphique. Les graphes de Tanner [38] sont particulièrement adaptés pour établir les relations entre les éléments c_n d'un mot de code.

La FIGURE 1.4 fournit la représentation d'une matrice de parité H de petite taille (FIGURE 1.4a) et son graphique de Tanner associé (FIGURE 1.4b). Les informations sur les éléments c_n d'un mot de code sont représentées par des cercles sur la partie supérieure de la graphe. Ils sont nommés *nœuds de variables*. Les contraintes sont représentées par des carrés sur la partie inférieure du graphe. On les dénomme *nœuds de contraintes*. Lorsque le graphe de Tanner représente un code lié à une matrice de parité, les nœuds de contraintes représente des équations de parités e_m. L'élément e_m référence aussi bien l'équation de parité indicée par m que son nœud de contrainte associé.



(a) (b)

FIGURE 1.4: Matrice de parité (a) et graphe de Tanner associé (b)

Le graphe de Tanner [38] permet de représenter les conditions de parité pour une matrice de parité associée à un code LDPC. Pour ces matrices, on introduit également la notion de *cycle*. Un cycle représente un ensemble fini de nœuds connectés débutant et terminant sur le même nœud. Chaque nœud est employé au maximum une fois dans le cycle. La taille du cycle correspond au nombre de nœuds traversés par ce cycle. Pour les codes LDPC, on s'intéresse aux cycles les plus courts. La taille minimale du cycle est appelé *girth*. La FIGURE 1.4b donne un exemple de cycle dont le chemin d'un *girth* est représenté en traits mixtes. Les cycles montrent les corrélations entre les différentes variables d'un mot de code LDPC. De ce fait, le décodage est d'autant plus efficace si ces équations sont décoréllées, et donc que le *girth* est grand.

1.3.1.3 Les codes QC-LDPC

La structure des codes LDPC est primordiale pour permettre d'atteindre de bonnes performances de décodage BER. Cependant, l'encodage et le décodage des

codes LDPC sont des tâches complexes si ceux-ci ne sont pas préalablement structurés. Dans la pratique, le nombre de chemins dans le graphe de Tanner augmente la complexité de calcul des informations de parité à l'encodage, et la structure de l'algorithme de décodage. Les standards de communication prévoient de nombreuses tailles de blocs et de rendements en fonction de la qualité du canal de transmission. Or, chaque usage spécifié sous la forme de **schéma de modulation et de codage** (MCS) nécessite de définir une matrice LDPC particulière. La diversité des usages requiert donc de structurer les codes LDPC afin de réduire la complexité globale.

Les codes QC-LDPC [13] forment une famille de codes structurés permettant de limiter cette complexité. Ils sont définis à travers une matrice identité carré de taille $z \times z$ que l'on note dans la suite I_0. À partir de cette matrice, on définit $z-1$ matrices notés I_j dont le motif correspond à la matrice I_0 permutée de j rangs vers la droite. D'un point de vue mathématique, cette permutation s'exprime par (1.21).

$$I_j = I_0 \times \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & & & & 1 \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}_{z\times z}^{j} \tag{1.21}$$

Une matrice QC-LDPC est donc définie comme une matrice bloc composée de matrices I_j ou de matrices carrées nulles 0_z (représentées par $-$). z désigne le **facteur d'expansion** de la matrice. Par construction, la dimension de la matrice H est un multiple de z. La FIGURE 1.5 illustre une matrice QC-LDPC provenant du standard IEEE 802.11n, de taille $M = 324$, $N = 648$ et $z = 27$. En réutilisant les notations précédemment établies, les matrices QC-LDPC sont aisément représentées.

I_0	-	-	-	I_0	I_0	-	-	I_0	-	-	I_0	I_1	I_0	-	-	-	-	-	-	-	-	-	-
I_{22}	I_{17}	-	-	I_0	-	I_0	I_0	I_{12}	-	-	-	-	I_0	I_0	-	-	-	-	-	-	-	-	-
I_6	-	I_0	-	I_{10}	-	-	-	I_{24}	-	I_0	-	-	-	I_0	I_0	-	-	-	-	-	-	-	-
I_2	-	-	I_0	I_{20}	-	-	-	I_{25}	I_0	-	-	-	-	-	I_0	I_0	-	-	-	-	-	-	-
I_{23}	-	-	-	I_3	-	-	-	I_0	-	I_9	I_{11}	-	-	-	-	I_0	I_0	-	-	-	-	-	-
I_{24}	-	I_{23}	I_1	I_{17}	-	I_3	-	I_{10}	-	-	-	-	-	-	-	-	I_0	I_0	-	-	-	-	-
I_{25}	-	-	-	I_8	-	-	-	I_7	I_{18}	-	-	I_0	-	-	-	-	-	I_0	I_0	-	-	-	-
I_{13}	I_{24}	-	-	I_0	-	I_8	-	I_6	-	-	-	-	-	-	-	-	-	-	I_0	I_0	-	-	-
I_7	I_{20}	-	I_{16}	I_{22}	I_{10}	-	-	I_{23}	-	-	-	-	-	-	-	-	-	-	-	I_0	I_0	-	-
I_{11}	-	-	-	I_{19}	-	-	-	I_{13}	-	I_3	I_{17}	-	-	-	-	-	-	-	-	-	I_0	I_0	-
I_{25}	-	I_8	-	I_{23}	I_{18}	-	I_{14}	I_9	-	-	-	-	-	-	-	-	-	-	-	-	-	I_0	I_0
I_3	-	-	-	I_{16}	-	-	I_2	I_{25}	I_5	-	-	I_1	-	-	-	-	-	-	-	-	-	-	I_0

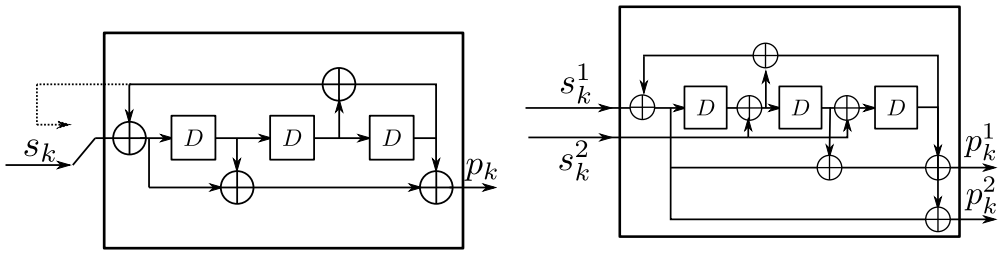
FIGURE 1.5: Matrice de parité d'un code QC-LDPC

Les matrices QC-LDPC constituées de matrices identités permutées partagent des propriétés architecturales intéressantes. L'ensemble des équations de parité représenté dans l'ensemble $[\![m.z; (m+1).z[\![$ forme un sous-code $\mathcal{C}^{\mathbf{m}}$ où chaque variable

n'est employée qu'au plus dans une seule équation. Ceci indique que chaque variable est localement indépendante des autres. De même chaque équation définie dans $\mathcal{C}^{\mathbf{m}}$ partage alors le même degré de parité. Ces propriétés permettent d'améliorer le parallélisme de l'architecture sans modifier les performances de décodage. Ces propriétés sont exploitées tout au long de ce mémoire.

1.3.2 Les Turbocodes Convolutifs

1.3.2.1 Les Codes Convolutifs



(a) (b)

FIGURE 1.6: Codes Convolutifs Récursifs et Systématiques binaire (a) et double-binaire (b)

Les codes convolutifs ont été introduits en 1955 par P. Elias [39] comme une alternative aux codes en blocs existants. Partant de la théorie de Shannon comme quoi un code de longueur infinie permet d'atteindre les limites théoriques, Elias cherche à définir une famille de codes de longueur variable. Le codage convolutif consiste à effectuer une opération de convolution sur un train d'informations binaires. Pour cela, le message $\boldsymbol{d}_1^K$ est regroupé en ensembles de m éléments notés s_k^m. Le codage convolutif renvoie n éléments en sortie notés p_k^n. Ces éléments de sorties sont déterminés à l'aide d'un ensemble $\boldsymbol{H}(z)$ de fonctions de convolution notées $\boldsymbol{H}^{\mathbf{n}}(z)$ exécutées sur les ν séquences de m éléments codés.

Cette transformée en z fait intervenir un rapport entre deux polynômes $\boldsymbol{G}^{\mathbf{n}}(z)$ et $\boldsymbol{N}(z)$ de degré maximal ν. Ceci permet de définir les diagrammes représentés sur la FIGURE 1.6. Toute transformée en z du code prend la forme de l'équation (1.22). La forme des polynômes composant les transformées en z permet de catégoriser le code convolutif.

$$\boldsymbol{G}^{\mathbf{i}}(z) = \sum_{j=0}^{\nu-1} g_j^{\mathbf{i}}.z^{-j} \tag{1.22}$$

Un code convolutif est dit **systématique** lorsque le bloc d'entrée de m bits se retrouve dans la séquence des n bits de sortie. Cette caractéristique se retrouve dans la forme de la matrice $\boldsymbol{H}(z)$. Cela signifie que l'un des polynômes de $\boldsymbol{H}(z)$ est unitaire. Un code convolutif systématique est un code systématique au sens de la définition 1.

Un code convolutif est dit **récursif** lorsque les polynômes de la matrice $\boldsymbol{H}(z)$ sont des quotients de polynômes. Dans ce cas, le code convolutif fait intervenir une boucle de rétroaction définie par le polynôme dénominateur $\boldsymbol{N}(z)$. Dans la pratique, ce polynôme dénominateur est commun à chaque polynôme de $\boldsymbol{H}(z)$.

Les dernières recherches montrent que pour des codes convolutifs de mêmes caractéristiques (m, n, μ), les meilleurs performances de décodage BER sont obtenues avec des codes convolutifs récursifs et systématiques alors noté codes RSC. Ces polynômes sont de la forme (1.23).

$$\boldsymbol{H}(z) = \left[1,\ \frac{\boldsymbol{G^1}(z)}{\boldsymbol{N}(z)}, \frac{\boldsymbol{G^2}(z)}{\boldsymbol{N}(z)},\ ...\right] \tag{1.23}$$

Le code RSC associé à l'opération (1.24) (respectivement (1.25)) est représenté sur la FIGURE 1.6a (resp. 1.6b).

$$\boldsymbol{H}(z) = \left[\frac{1 + z^{-1} + z^{-3}}{1 + z^{-2} + z^{-3}}\right] \tag{1.24}$$

$$\boldsymbol{H}(z) = \left[\frac{1 + z^{-2} + z^{-3}}{1 + z^{-1} + z^{-3}},\ \frac{1 + z^{-3}}{1 + z^{-1} + z^{-3}}\right] \tag{1.25}$$

1.3.2.2 Représentation en treillis

Les codes convolutifs font intervenir ν éléments de mémoire D, dont les valeurs sont notées M_1^k, ... M_ν^k à un instant k donné. On définit l'état du système par la variable S_k. Cet état est déterminé en fonction de la valeur de ces mémoires à l'instant k. S_k vérifie l'équation (1.26).

$$S_k = \sum_{j=0}^{\nu-1} 2^j . M_{\nu-j}^k \tag{1.26}$$

À partir du schéma de représentation d'un code convolutif, il advient que les sorties p_k^n du code convolutif sont définies intégralement à partir de la connaissance de la valeur des registres et de l'entrée du système. Le code convolutif respecte de ce fait la propriété 1.3.1 de Markov.

Proposition 1.3.1. *L'état de la machine associée à un processus markovien à l'instant $k+1$ ne dépend que de l'entrée de cette machine et de l'état courant de celle-ci. Ceci se traduit mathématiquement par l'équation (1.27).*

$$\Pr\{S_{k+1} = s | S_k = s_k, S_{k-1} = s_{k-1}, ..., S_0 = s_0\} = \Pr\{S_{k+1} = s | S_k = s_k\} \tag{1.27}$$

L'intégralité des entrées et des parités associées est connue à partir de la connaissance de l'ensemble des états S_k du codeur. Un diagramme **en treillis** représentant l'ensemble des transitions possibles entre les états d'un codeur permet alors de représenter l'ensemble des étapes du codage. Sur ce type de graphique, un mot de code

est représenté par un chemin unique sur ce treillis. L'ensemble de l'encodage de K éléments binaires définit un treillis de longueur K/m. La FIGURE 1.7 représente dans ce cas un chemin de treillis parcouru par un mot de code spécifique.

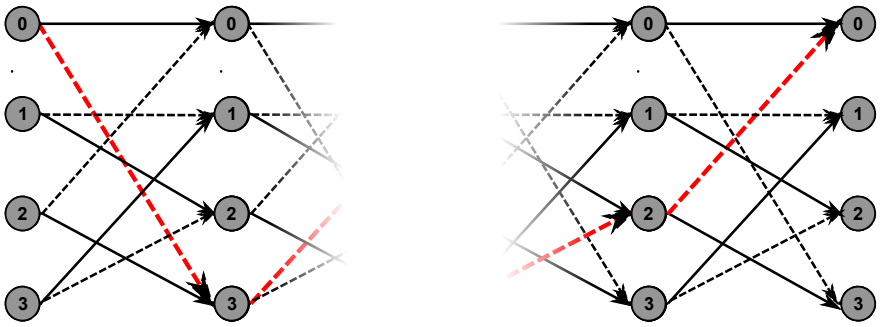


FIGURE 1.7: Un chemin de treillis parcouru par un mot de code

L'ensemble des transitions possibles entre deux états de la machine d'état décrit une section sur ce treillis. Ce motif de transition se répète sur chaque section de treillis. La FIGURE 1.8 représente deux sections du treillis associées au code convolutif (1.24).

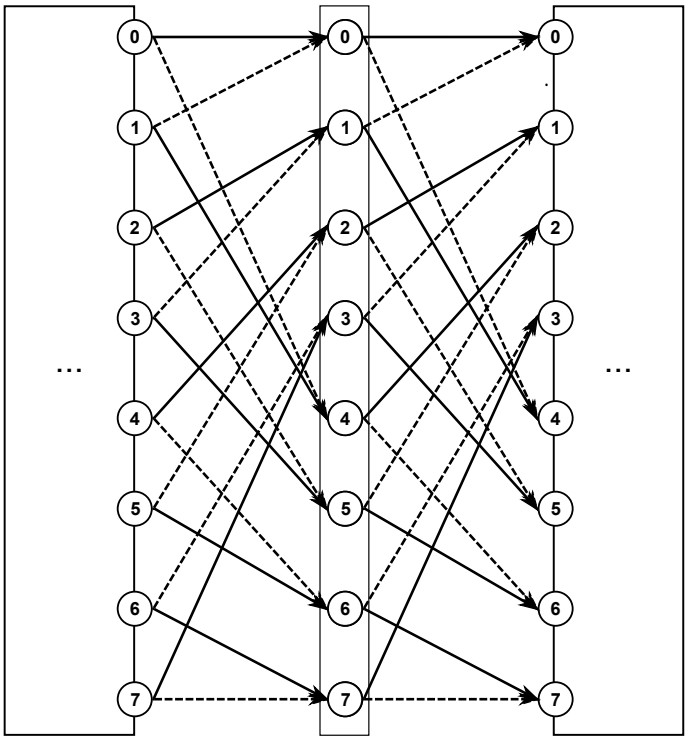


FIGURE 1.8: Deux sections de treillis associés au code convolutif (1.24)

En pratique, les codes convolutifs sont des codes de taille finie. Cependant il est possible d'obtenir de meilleures performances de décodage si l'état en fin de treillis

est connu. En pratique, on applique des techniques de fermeture de treillis sur des codes convolutifs récursifs afin qu'état initial et état final soient identiques. De ce fait, le treillis du code convolutif peut se refermer sur lui-même. Deux techniques de fermeture de treillis sont couramment employées.

La technique de Retour à zéro (ou *Zero-Forcing*) [40] nécessite d'ajouter ν sections à la fin du treillis. Pour cela, les informations systématiques sont constituées par les valeurs de la récursion. Par cette technique, le treillis se retrouve systématiquement à l'état 0 pour la section $k = K/m+\nu$. Cette technique permet un encodage rapide, mais nécessite l'envoi de ν bits systématiques supplémentaires ainsi que les bits de redondance associés, ce qui entraîne une perte d'efficacité spectrale.

Les codes RSC circulaires [41] (CRSC) ont été introduits pour éviter cette perte d'efficacité spectrale. Les codes CRSC contraignent la matrice génératrice du code convolutif de sorte que pour une taille de bloc K définie, il existe un état de circulation s_c tel que l'encodage du bloc commence et se termine à cet état. L'état de circulation est donc conditionné à la structure du code, mais également à la taille du bloc K. En cas d'existence de cette état, celui-ci est fonction de l'état final du codeur débutant à l'état initial connu '0' noté s_f et de la taille du bloc K/m. Cette relation est donc de la forme (1.28).

$$s_c = f(K/m, s_f) \tag{1.28}$$

Pour trouver cet état de circulation, il est donc nécessaire d'encoder une première fois le message, sans prendre en compte les informations de redondance. L'état final s_f permet de définir un état de circulation en fonction de la taille du message émis. Cette technique nécessite d'encoder chaque message deux fois, ce qui complexifie l'encodage des paquets à émettre. Elle évite d'émettre les bits supplémentaires nécessaires pour effectuer un retour à l'état 0, et permet donc d'améliorer l'efficacité spectrale du code et de garantir une protection des données uniformément répartie. L'état s_c n'est pas connu au niveau du décodage.

La fermeture de treillis permet d'éviter les effets de bords liés à ce manque d'information. A la réception, l'état initial et l'état final sont identiques. De ce fait, les probabilités liés à ces états en fin de treillis et en début de treillis sont identiques. Cette notion permet de représenter le schéma de treillis sur un cercle en joignant la section initiale et la section finale. Cette représentation permet de modifier le processus de décodage et d'améliorer la connaissance de cet état s_c. La section 3.3.1 revient sur cette notion. La FIGURE 1.9 représente le treillis d'un code CRSC fermé.

1.3.2.3 Les Turbocodes Convolutifs

Les Codes Convolutifs sont utilisés dans de nombreux standards. Leur longueur indéfinie permet un usage flexible de codage. De plus, en augmentant le nombre d'états, ces codes permettent d'atteindre des performances de décodage proches de quelques décibels de la limite de Shannon. L'introduction du principe de décodage itératif permet de s'approcher plus encore de cette limite.

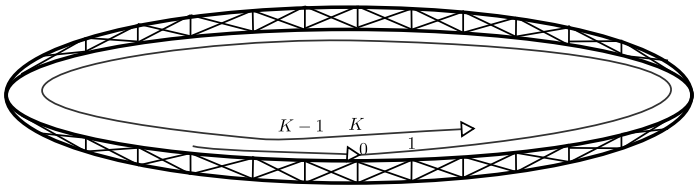


FIGURE 1.9: Fermeture de treillis pour un code CRSC

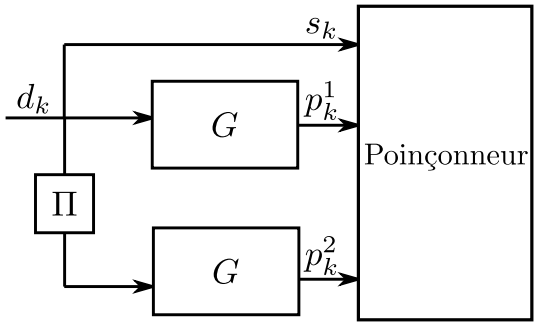


FIGURE 1.10: Encodage des turbocodes convolutifs

Les turbocodes convolutifs (notés abusivement turbocodes dans la suite) représentent une famille de codes constituée par la mise en parallèle de deux codes convolutifs notés $\mathcal{C}_1$ et $\mathcal{C}_2$ et séparés par un entrelaceur Π. Ce type d'encodage, aussi nommé encodage PCCC (*Parallel concatened convolutional codes*) est représenté sur la FIGURE 1.10. En pratique, les deux codes constituants $\mathcal{C}_1$ et $\mathcal{C}_2$ sont équivalents et de type RSC [10].

Les turbocodes utilisent en entrée le message à émettre $\boldsymbol{d}_1^K$. Lorsque le nombre d'entrées m du code constituant vaut 1, on parle de turbocodes binaires. Les entrées des codes constituants sont alors représentées sous la forme s_k. Les turbocodes double-binaires utilisent des codes constituants avec $m = 2$. Le message source est divisé en couples de données binaires représentés par $\left(s_k^1, s_k^2\right)$.

Le rendement naturel d'un turbocode est donné par le rendement global du code. Des rendements supérieurs sont obtenus en poinçonnant l'information de redondance en sortie du codeur. Les bits supprimés suivent un motif périodique spécifique à chaque rendement. Cette technique est employée pour améliorer l'efficacité des codes convolutifs et a été étudiée pour le cas des turbocodes par [42].

La grande force des turbocodes réside dans le choix de son entrelacement Π. L'entrelaceur permet d'augmenter la distance minimale du code constituant. Un entrelaceur aléatoire [42] permet d'éviter toute corrélation entre les deux décodeurs parallèles. En pratique cependant, les lois d'entrelacement doivent pouvoir être calculées facilement de manière à réduire les contraintes de mise en œuvre. Les entrelaceurs **réguliers** [43] ne permettent pas de casser les cycles d'erreurs et réduisent les performances du code. Les entrelaceurs de type ARP(*Almost Regular Permuta-*

tion) [44] comme (1.29) font intervenir une fonction cyclique $Q(j)$. Ces entrelaceurs permettent un calcul à la volée des fonctions d'entrelacement tout en augmentant la distance minimale du turbocode.

Les entrelaceurs QPP (*Quadratic Polynomial Permutation*) [45] comme (1.30) offrent une irrégularité permettant de meilleurs performances de décodage. Dans ces expressions, P_0 et P_1 sont des constantes définies par le code, et $Q(j)$ est une fonction cyclique.

$$\Pi(k) = (P_1.k + Q(k) + P_0)_{[K/m]} \tag{1.29}$$

$$\Pi(k) = \left(P_1.k^2 + P_0.k\right)_{[K/m]} \tag{1.30}$$

Les entrelaceurs ARP et QPP sont sélectionnés dans les standards actuels. D'autres fonctions d'entrelacements ont été étudiées pour améliorer les performances de décodage comme l'entrelacement DRP (*Dithered Relative Prime*) [46] mais ne sont pas intégrées dans les codes standardisés actuellement.

1.4 Décodage de codes correcteurs d'erreurs avancés

Les grandes caractéristiques de codes correcteurs d'erreurs avancés sont détaillées dans la section 1.3 de ce chapitre. Le décodage de ces codes nécessite de recourir à des algorithmes de décodage itératifs. Cette section détaille le principe du décodage de canal et analyse les stratégies de décodage suivant la représentation des codes associés.

1.4.1 Principe du décodage de canal

Les codes correcteurs d'erreurs sont comparés en fonction de leurs caractéristiques (N, K, d_{min}). Le choix de l'algorithme de décodage est également un critère de choix pour atteindre de bonnes performances en termes de BER ou de FER.

Le décodage maximisant le taux d'erreurs trame FER nécessite de définir des algorithmes de décodage vérifiant l'équation (1.31).

$$\hat{\boldsymbol{d}}_1^K = \underset{\boldsymbol{d}_1^K \in \mathcal{C}}{\arg\max} \left(\Pr \left\{ \boldsymbol{d}_1^K | \boldsymbol{y}_1^{N_m} \right\} \right) \tag{1.31}$$

Ce type de décodage n'est pas optimal pour des codes de grandes longueurs, puisqu'une erreur de décision entraîne nécessairement un minimum de d_{min} bits erronés sur la trame décidée. Un autre type de décodage revient à minimiser le taux d'erreurs binaire (BER). Ce décodage est nommé décodage du Maximum *A Posteriori* (MAP), et est formalisé par l'équation (1.32).

$$\hat{d}_k = \underset{d_k \in \mathcal{A}}{\arg\max} \left(\Pr \left\{ d_k | \boldsymbol{y}_1^{N_m} \right\} \right) \tag{1.32}$$

Lorsque la répartition des évènements est connue, il est préférable de prendre une décision suivant le maximum de vraisemblance (ML) défini par la fonction (1.33) dans le cas d'une minimisation du BER.

$$\hat{d}_k = \underset{d_k \in \mathcal{A}}{\arg\max} \left(\Pr \left\{ \boldsymbol{y}_1^{N_m} | d_k \right\} \right) \tag{1.33}$$

Les opérations (1.32) et (1.33) sont équivalentes lorsque les évènements sont équiprobables. Ceci implique que la source émette des données indépendantes et identiquement distribuées (i.i.d.).

Les performances de décodage sont contraintes par le type de décodeur. Un algorithme de décodage dur utilise en entrée les décisions dures du démodulateur $\hat{\boldsymbol{c}}_1^N$. Dans le cadre du décodage de codes correcteurs d'erreurs avancés, l'algorithme de Viterbi [47] permet de décoder les codes convolutifs. L'algorithme A de Gallager [11] s'adresse aux codes LDPC.

Cependant, pour des canaux de transmission à sortie réelles comme le canal AWGN, les décodeurs à décision dure sont sous-optimaux. La prise en compte de l'intégralité de l'information nécessite des algorithmes de décodage à décision souple. L'entrée de ce type de décodage prend en compte des probabilités d'évènements, les rapports de vraisemblance ou leurs représentations logarithmiques. Le Log-Rapport de Vraisemblance en sortie du canal est défini par la relation (1.3). Sur le même principe, on définit le Log-Rapport de Vraisemblance *a posteriori* $\mathrm{L}^a(c_k)$ selon (1.34) et le Log-Rapport de Vraisemblance extrinsèque comme (1.35). Ces trois définitions sont reliées par la relation (1.36) qui s'obtient en appliquant la règle de Bayes.

$$\mathrm{L}^a(c_k) \triangleq \log \left(\frac{\Pr \left\{ c_k = 1 | \boldsymbol{y}_1^{N_m} \right\}}{\Pr \left\{ c_k = 0 | \boldsymbol{y}_1^{N_m} \right\}} \right) \tag{1.34}$$

$$\mathrm{L}^e(c_k) \triangleq \log \left(\frac{\Pr \left\{ c_k = 1 | \boldsymbol{y}_1^{k-1}, \boldsymbol{y}_{k+1}^{N_m} \right\}}{\Pr \left\{ c_k = 0 | \boldsymbol{y}_1^{k-1}, \boldsymbol{y}_{k+1}^{N_m} \right\}} \right) \tag{1.35}$$

$$\mathrm{L}^a(c_k) = \mathrm{L}^c(c_k) + \mathrm{L}^e(c_k) \tag{1.36}$$

À partir des Log-Rapports de Vraisemblance, il devient possible de prendre une décision dure sur les variables, en appliquant la relation (1.37).

$$\mathrm{L}^a(c_k) > 0 \Leftrightarrow \Pr \left\{ c_k = 1 | \boldsymbol{y}_1^{k-1}, \boldsymbol{y}_{k+1}^{N_m} \right\} > \Pr \left\{ c_k = 0 | \boldsymbol{y}_1^{k-1}, \boldsymbol{y}_{k+1}^{N_m} \right\} \tag{1.37}$$

Dans ce cas, le décodage MAP maximisant le BER revient à prendre une décision en fonction du signe du Log-Rapport de Vraisemblance.

1.4.2 Décodage d'équations de parités

Une contrainte de parité est une équation qui vérifie la relation (1.38).

$$c_1 \oplus c_2 \oplus ... \oplus c_{d_c} = 0 \tag{1.38}$$

L'opérateur $\oplus$ exprime une opération binaire XOR. L'opération (1.38) est équivalente à la relation (1.39). Celle-ci permet d'isoler un opérande de l'expression.

$$c_k = c_1 \oplus ... \oplus c_{k-1} \oplus c_{k+1} \oplus ... \oplus c_{d_c} \tag{1.39}$$

L'élément étudié c_k est entièrement conditionné par les autres opérandes de l'expression (1.39). On s'intéresse à connaître la probabilité de l'élément conditionnellement aux opérandes. Cette étude permet d'établir une décision sur cet élément en appliquant le principe de décodage MAP. La proposition 1.4.1 est une étape préliminaire à l'établissement de cette probabilité d'apparition.

Proposition 1.4.1. *Pour une équation de parité vérifiant (1.40), il est possible d'établir la relation (1.41)*

$$e_n = c_1 \oplus c_2 \oplus ... \oplus c_n \tag{1.40}$$

$$\tanh\left(\frac{1}{2} \times \log\left(\frac{\Pr\left(e_n = 1 | c_1, ..., c_n\right)}{\Pr\left(e_n = 0 | c_1, ..., c_n\right)}\right)\right) = \sum_{i=1}^{n} \tanh\left(\frac{1}{2} \times \log\left(\frac{\Pr\left(c_i = 1\right)}{\Pr\left(c_i = 0\right)}\right)\right) \tag{1.41}$$

Cette proposition est démontrée dans l'Annexe C de ce mémoire. Elle permet de conditionner chaque élément d'une équation de parité en fonction des autres opérandes. En replaçant cette expression dans le contexte du décodage de matrices de parité, et avec les notations précédemment établies, d'une équation de parité e_m faisant intervenir les éléments dont les indices sont contenus dans l'ensemble $\mathcal{N}_m$, cette expression devient (1.42).

$\forall c_n \in \mathcal{N}_m$,

$$\tanh\left(\frac{1}{2}.\log\left(\frac{\Pr\left\{c_n = 1 | c_{n'} \in \mathcal{N}_{m|n}\right\}}{\Pr\left\{c_n = 10 | c_{n'} \in \mathcal{N}_{m|n}\right\}}\right)\right) = \sum_{n' \in \mathcal{N}_{m|n}} \tanh\left(\frac{1}{2}.\log\left(\frac{\Pr\left\{c_{n'} = 1\right\}}{\Pr\left\{c_{n'} = 0\right\}}\right)\right) \tag{1.42}$$

Afin de simplifier cette expression, on définit la fonction ϕ par l'expression (1.43) qui est représentée FIGURE 1.11.

$\forall x \in \mathbb{R}_+^*$,

$$\phi(x) = \phi^{-1}(x) = -\log\left(\tanh\frac{x}{2}\right) \tag{1.43}$$

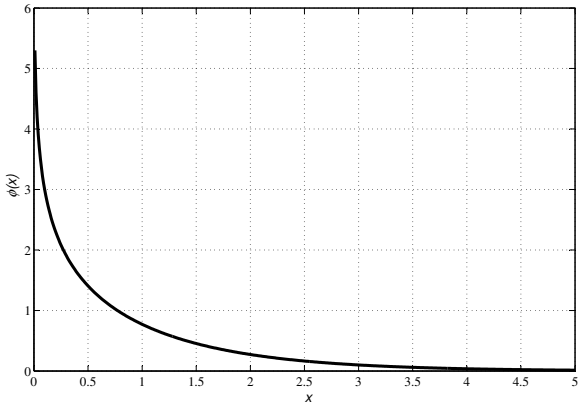


FIGURE 1.11: Représentation de la fonction ϕ

Cette fonction est inversible et son inverse est égal à elle même. De ce fait, en prenant en compte la valeur absolue du Log-Rapport de Vraisemblance attribuée à chaque élément, l'équation (1.45) permet de déduire la confiance attribuée à chaque variable d'une équation de parité. La décision dure est établie par l'expression (1.44).

$$\operatorname{sgn}\left(\mathrm{L}^{e}\left(c_{n}\right)\right) = \bigoplus_{n' \in \mathcal{N}_{m}, n' \neq n} c_{n'} \tag{1.44}$$

$$\left|\mathrm{L}^{e}\left(c_{n}\right)\right| = \phi^{-1}\left(\sum_{n \in \mathcal{N}_{m|n}} \phi\left(\left|\mathrm{L}^{a}\left(c_{n'}\right)\right|\right)\right) \tag{1.45}$$

Cette expression est cependant complexe à calculer. De nombreuses pistes ont été explorées afin de limiter cette complexité. [48] propose de ne prendre en compte que les λ valeurs les moins fiables pour chaque équation de parité, [49] approche cette opération par un minimum, dont la complexité est moindre mais dégrade fortement les performances de décodage. [50] corrige cette approximation en introduisant des constantes de correction additif (OMS) ou multiplicatif (NMS). La TABLE 1.3 donne les expressions analytiques de ces approximations.

1.4.3 Décodage par représentation sur graphe de Tanner

Le décodage par propagation de croyance est un algorithme de décisions souples permettant d'approcher une décision MAP. Il se base sur le transfert de messages de confiance entre des nœuds de contraintes et des nœuds de variables et se représente sur un graphe de Tanner. La FIGURE 1.12 donne un aperçu des messages traités par un nœud de contrainte e_1 et renvoyés à la variable c_3. On définit $\mathrm{L}_{m}^{a}\left(c_{n}\right)$ la confiance attribuée à l'élément c_n envoyée vers la contrainte e_m et $\mathrm{L}_{m}^{e}\left(c_{n}\right)$ la confiance renvoyée vers l'élément c_n relativement à l'information calculée par le nœud de contrainte e_m.

Dénomination	Approximation
Sum-Product (SPA) [11]	$\left\|\mathrm{L}^{e}\left(c_{n}\right)\right\| = \phi^{-1}\left(\sum_{n\in\mathcal{N}_{m\,\vert n}} \phi\left(\left\|\mathrm{L}^{a}\left(c_{n'}\right)\right\|\right)\right)$ (1.46)
λ-Min (λM) [48]	$\left\|\mathrm{L}^{e}\left(c_{n}\right)\right\| = \phi^{-1}\left(\sum_{n\in\mathcal{N}_{m\,\vert n}^{\lambda}} \phi\left(\left\|\mathrm{L}^{a}\left(c_{n'}\right)\right\|\right)\right)$ (1.47)
Min-Sum (MS) [49]	$\left\|\mathrm{L}^{e}\left(c_{n}\right)\right\| = \min_{c_{n'\neq n}\in\mathcal{N}_{m}} \left\|\mathrm{L}^{a}\left(c_{n'}\right)\right\|$ (1.48)
Offset Min-Sum (OMS) [50]	$\left\|\mathrm{L}^{e}\left(c_{n}\right)\right\| = \min_{c_{n'\neq n}\in\mathcal{N}_{m}} \left\|\mathrm{L}^{a}\left(c_{n'}\right)\right\| + C^{te}$ (1.49)
Normalized Min-Sum (NMS) [50]	$\left\|\mathrm{L}^{e}\left(c_{n}\right)\right\| = C^{te} \times \min_{c_{n'\neq n}\in\mathcal{N}_{m}} \left\|\mathrm{L}^{a}\left(c_{n'}\right)\right\|$ (1.50)

TABLE 1.3: Approximations du décodage pour une équation de parité

Dans ce cas, le calcul de la confiance $\mathrm{L}_m^a(c_n)$ fait référence à une étape de **mise à jour des nœuds de parité**. De même, le calcul de la confiance $\mathrm{L}_m^e(c_n)$ est effectuée lors de la **mise à jour des nœuds de variable**.

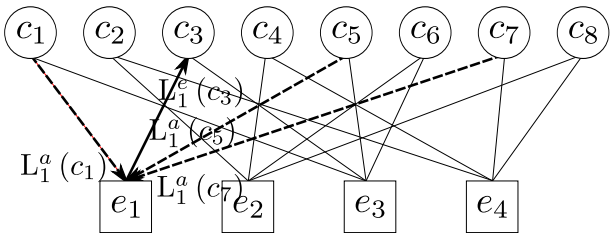


FIGURE 1.12: Transfert d'information sur un graphe de Tanner

Il reste encore à définir la nature des messages transférés entre chaque nœud. La connaissance *a priori* attribuée à chaque élément du mot de code c_i est connue et repérée par l'expression $\mathrm{L}^c(c_i)$.

Dans un premier temps, on cherche à définir l'information descendante $\mathrm{L}_m^a(c_n)$ en fonction de la connaissance de l'ensemble des conditions du code. Pour cela, on suppose que l'ensemble des informations remontant des contraintes $\mathrm{L}_m^e(c_n)$ sont indépendantes de la variable c_n. La véracité de ce présupposé est conditionnée à la forme de la matrice LDPC. Dans ce cas, l'équation (1.51) est vérifiée.

$$\Pr\left\{c_n = i|\boldsymbol{y}_1^{N_m}\right\} = \Pr\left\{c_n = i \left| y_n, \bigoplus_{n'\in\mathcal{N}_{1|n}} c_{n'} = i, ..., \bigoplus_{n'\in\mathcal{N}_{M|n}} c_{n'}\right.\right\} \tag{1.51}$$

Cette quantité s'obtient avec un Log-Rapport de vraisemblance à travers la relation (1.52).

$$\mathrm{L}^a(c_n) = \log\left(\frac{\Pr\left\{c_n = 1|\boldsymbol{y}_1^{N_m}\right\}}{\Pr\left\{c_n = 0|\boldsymbol{y}_1^{N_m}\right\}}\right) \tag{1.52}$$

En remplaçant l'expression (1.51) dans la définition du Log-Rapport de Vraisemblance, il advient l'expression (1.53).

$$\mathrm{L}^a(c_n) = \underbrace{\log\left(\frac{\Pr\{c_n = 1|y_n\}}{\Pr\{c_n = 0|y_n\}}\right)}_{\mathrm{L}^c(c_n)} + \sum_{m=1}^{M} \underbrace{\log\left(\frac{\Pr\left\{c_n = 1|c_{n'}\in\mathcal{N}_{m|n}\right\}}{\Pr\left\{c_n = 0|c_{n'}\in\mathcal{N}_{m|n}\right\}}\right)}_{\mathrm{L}_m^e(c_n)} \tag{1.53}$$

La métrique $\mathrm{L}^a(c_n)$ est dépendante des données fournies par le nœud de contrainte e_m. Pour éviter les risques d'auto-confirmation, il est inévitable de renvoyer à ce nœud cette information privée de l'information provenant de ce nœud. Mathématiquement, ceci correspond à définir $\mathrm{L}_m^a(c_n)$ selon (1.54) qui équivaut à (1.55).

$$\mathrm{L}^a_m(c_n) = \mathrm{L}^c(c_n) + \sum_{m' \in \mathcal{M}_{n|m}} \mathrm{L}^e_{m'}(c_n) \tag{1.54}$$

$$\mathrm{L}^a_m(c_n) = \mathrm{L}^a(c_n) - \mathrm{L}^e_m(c_n) \tag{1.55}$$

L'algorithme de propagation de croyance consiste à transférer des informations $\mathrm{L}^e_m(c_n)$ obtenues des nœuds de contrainte vers les nœuds de variable puis de mettre à jour cette information afin de retransmettre l'information actualisée $\mathrm{L}^a_m(c_n)$ aux nœuds voisins.

L'enchaînement des opérations de mise à jour a été longuement étudié. [11] définit un ordonnancement par inondation ou *flooding*. Le décodage est divisé en deux étapes successives, dans lesquelles tous les nœuds de variable sont mis à jour, puis tous les nœuds de contrainte. On définit dans ce cas une itération comme la mise à jour successive de chaque nœud de variable et de chaque nœud de parité. Cet ordonnancement est illustré sur la FIGURE 1.13.

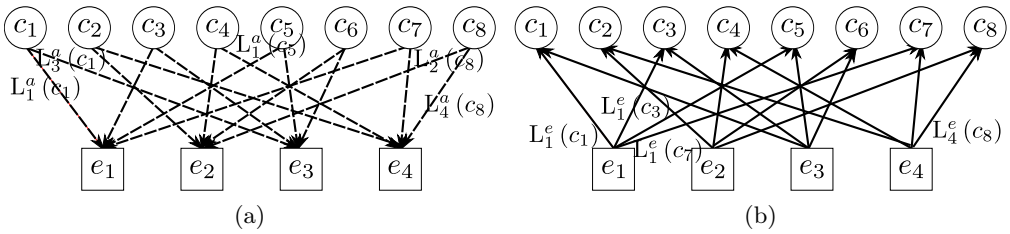


FIGURE 1.13: Première (a) et deuxième (b) étape de mise à jour par inondation

Cependant les performances de décodage de cet ordonnancement sont fortement impactées par la structure de la matrice de parité. L'information $\mathrm{L}^e_m(c_n)$ n'est plus indépendante de la variable c_n lorsque le nombre d'itérations dépasse la moitié de la taille du cycle court du code LDPC. De ce fait, à partir d'un certain nombre d'itérations, les performances de décodage sont impactées par les phénomènes d'auto-confirmation.

[51] propose un ordonnancement des mises à jour probabiliste basé sur l'ordonnancement par inondation en stoppant les mises à jour des variables en fonction des tailles des cycles. Cet ordonnancement améliore les performances de décodage en évitant les phénomènes d'auto-confirmation mais requiert une étude spécifique à chaque matrice de parité.

Afin de réduire les phénomènes d'auto-confirmation ainsi que la complexité de l'algorithme par inondation, [52] propose un traitement par couches du décodage (ou ordonnancement *Layered Belief-Propagation*). L'ordonnancement en couches verticales propose d'exécuter une mise à jour successive sur les nœuds de variable et de recalculer les informations sur les nœuds de contrainte modifiés. [53] étudie un ordonnancement par couches noté Zigzag LBP qui modifie l'ordre de calcul incrémental de l'ordonnancement en couches verticales.

L'ordonnancement en couches horizontales (noté H-LBP pour *Horizontal Layered Belief-Propagation* dans la suite) propose la mise à jour des nœuds de contrainte un à un et de remonter directement l'information sur les nœuds de variable, comme illustré sur la FIGURE 1.14.

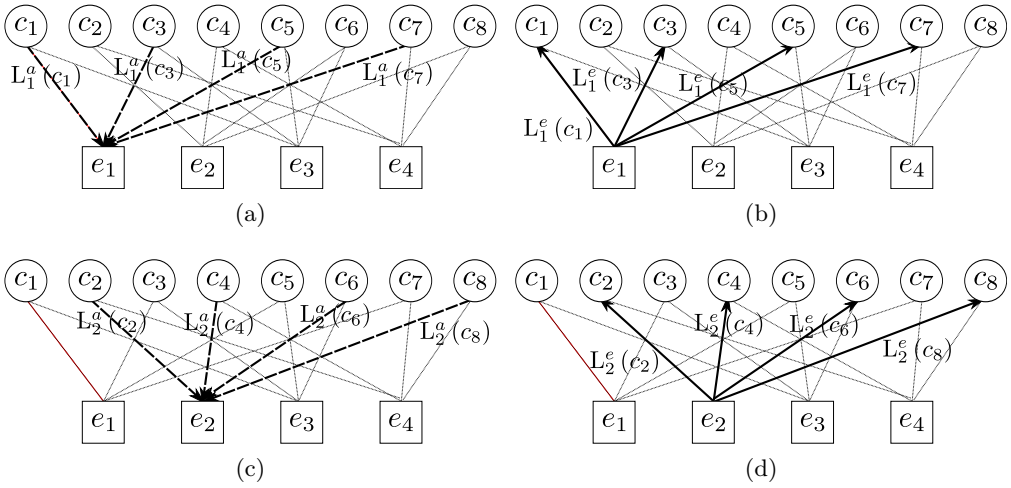


FIGURE 1.14: Première (a), deuxième (b), troisième (c) et quatrième (d) étape de mise à jour par couches horizontales

1.4.4 Décodage par représentation en treillis

En 1967, A. Viterbi [47] propose un premier algorithme de décodage de codes convolutifs basé sur un décodage dur suivant le principe du maximum de vraisemblance. Il est adapté au décodage de codes représentés en treillis. Cet algorithme est généralisé au décodage souple en 1989 avec l'algorithme SOVA de J. Hagenauer [54]. En 1974, L. Bahl, et al. propose un nouvel algorithme (Algorithme BCJR [55]) pour décoder des codes convolutifs suivant le maximum a posteriori (MAP) et minimisant la probabilité d'erreur binaire. En 1993, C. Berrou [10] s'inspire de l'algorithme de décodage BCJR et l'adapte au décodage itératif de codes convolutifs.

Dans la suite, on note $s_k = \left(s_k^1, s_k^2, ...\right)$ l'ensemble des éléments systématiques binaires relatifs à la section k du treillis. De même, on note $p_k = \left(p_k^1, p_k^2, ...\right)$ l'ensemble des éléments de redondance de cette section. Pour plus de simplicité, on considère que le treillis possède exactement K sections.

Le décodage BCJR recherche à minimiser le BER à partir d'une définition d'un code en treillis. D'après la propriété 1.3.1, la probabilité de la machine d'être à l'état $S_k = s'$ est entièrement déterminée par la connaissance des probabilités des états précédents à l'instant k et aux probabilités des événements s_k et p_k. De ce fait, la probabilité des éléments s_k est établie par la formule (1.56), qui reprend l'énoncé du décodage MAP présenté par (1.32).

$$\Pr\{s_k = i\} = \sum_{\substack{s \longrightarrow s' \\ s_k = i}} \underbrace{\Pr\left\{S_{k-1} = s, S_k = s'\right\}}_{\lambda_k(s,s')} \tag{1.56}$$

La quantité $\lambda_k(s, s')$ représente dans ce cas chaque opérande. Cette quantité se décompose suivant (1.57).

$$\begin{aligned} \lambda_k(s,s') = & \underbrace{\Pr\{s_1, p_1, ..., s_{k-1}, p_{k-1}, S_{k-1} = s\}}_{\alpha_{k-1}(s)} \\ \times & \underbrace{\Pr\left\{S_k = s', s_k, p_k | S_{k-1} = s\right\}}_{\gamma_k(s,s')} \\ \times & \underbrace{\Pr\left\{s_{k+1}, p_{k+1}, ..., s_K, p_K | S_k = s'\right\}}_{\beta_k(s')} \end{aligned} \tag{1.57}$$

Le premier terme correspond à la probabilité que le treillis soit à l'état s' à l'instant $k-1$ en connaissant l'ensemble de l'information sur les sections 0 à $k-1$ du treillis. Cette information passée du treillis est notée $\alpha_{k-1}(s)$ dans la suite. De même, le dernier terme correspond à la probabilité que le treillis soit à l'état s en connaissant l'ensemble des informations sur les sections $k+1$ à K du treillis. Elle est notée $\beta_k(s)$ dans la suite. Le terme intermédiaire noté $\gamma_k(s', s)$ correspond à la probabilité de passage de l'état s' à l'état s avec la seule connaissance de l'information de la section, soit l'information intrinsèque systématique s_k, l'information intrinsèque de parité p_k et une éventuelle information extrinsèque sur l'information systématique noté $\mathrm{L}^e(s_k)$. Ces données sont illustrées sur un schéma de treillis sur la FIGURE 1.15.

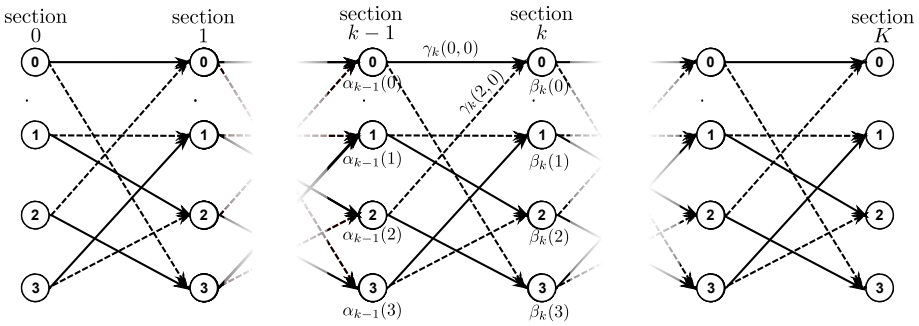


FIGURE 1.15: Représentation des métriques de calcul de l'algorithme BCJR sur un treillis

En reprenant les définitions des Log-Rapports de Vraisemblance de la partie 1.4.1, il advient (1.58).

$$\mathrm{L}^{a}\left(c_k\right) = \log\left(\frac{\sum\limits_{s \xrightarrow[c_k=1]{} s'} \alpha_{k-1}(s).\gamma_k(s,s').\beta_k(s')}{\sum\limits_{s \xrightarrow[c_k=0]{} s'} \alpha_{k-1}(s).\gamma_k(s,s').\beta_k(s')}\right) \tag{1.58}$$

Les informations concernant le passé (resp. le futur) du treillis sont calculées par itération à partir du début (resp. de la fin) du treillis suivant les relations (1.59) (resp. (1.60)).

$$\alpha_k(s') = \sum_{s} \alpha_{k-1}(s).\gamma_k(s,s') \tag{1.59}$$

$$\beta_k(s) = \sum_{s'} \beta_{k+1}(s').\gamma_{k+1}(s,s') \tag{1.60}$$

De ce fait, l'algorithme BCJR est souvent nommé un algorithme *Aller-Retour*. Le sens *Aller* permet de calculer les informations $\alpha_{k-1}(s)$, et le sens *Retour* les informations $\beta_k(s')$

L'information $\gamma_k(s,s')$ rassemble toute la connaissance sur une transition de treillis, soit l'information systématique, l'information de parité et l'information extrinsèque incidente en cas de décodage itératif. Cette quantité s'exprime par l'information du Log-Rapport de Vraisemblance sur les éléments s_k et p_k. En reprenant les notations introduites dans la section 1.4.1, la connaissance sur une transition de treillis s'obtient par la relation (1.61).

$$\gamma_k(s,s') = (-1)^{s_k^{s\to s'}}.\left(\mathrm{L}^{c}\left(s_k\right) + \mathrm{L}^{e}\left(s_k\right)\right) + (-1)^{p_k^{s\to s'}}.\mathrm{L}^{c}\left(p_k\right) \tag{1.61}$$

L'algorithme BCJR initial a été modifié par [10] afin de bénéficier d'un effet turbo relatif aux décodage de turbocodes. [10] définit une information extrinsèque en sortie $\mathrm{L}^{e,out}\left(s_k\right)$ suivant (1.62). Dans le cas de décodage itératif comme pour le décodage de turbocodes, cette information transite entre les différents décodeurs et évite les phénomènes d'auto-confirmation.

$$\mathrm{L}^{e,out}\left(s_k\right) = \log\left(\frac{\sum\limits_{s \xrightarrow[s_k=1]{} s'} \alpha_{k-1}(s).\Pr\left\{y_k|p_k^{s\to s'}\right\}.\beta_k(s')}{\sum\limits_{s \xrightarrow[s_k=0]{} s'} \alpha_{k-1}(s).\Pr\left\{y_k|p_k^{s\to s'}\right\}.\beta_k(s')}\right) \tag{1.62}$$

Le calcul des probabilités dans le corps logarithmique permet de transformer les multiplications en additions, et les additions en logarithmes jacobiens *logsum* ou $\max^*$ définis par les relations (1.63-1.64).

$$\begin{aligned} \max{}^*(x,y) &\triangleq \log(e^x + e^y) \\ \max{}^*(x,y) &= \max(x,y) + \log(1 + e^{-|x-y|}) \end{aligned} \tag{1.63}$$

$$\begin{aligned} \max{}^*(x_1,...,x_n) &\triangleq \log(e^{x_1}+...+e^{x_n}) \\ \max{}^*(x_1,...,x_n) &= \max(x_1,...,x_n)+\log(1+e^{-|x_m-x_1|}+...+e^{-|x_m-x_n|}) \end{aligned} \tag{1.64}$$

On note ici $x_m = max(x_1,...,x_n)$. Les opérations (1.59-1.60) sont simplifiés par les opérations (1.65-1.66).

$$\log\left(\alpha_k(s')\right) = \max_s{}^*\left(\alpha_{k-1}(s)+\gamma_k(s,s')\right) \tag{1.65}$$

$$\log\left(\beta_k(s)\right) = \max_{s'}{}^*\left(\beta_{k+1}(s')+\gamma_{k+1}(s,s')\right) \tag{1.66}$$

En prenant en compte l'opérateur $\max^*$, la relation (1.58) est équivalente à la relation (1.67) et (1.62) devient (1.68).

$$\begin{aligned} \mathrm{L}^a(c_k) = &\max_{\substack{s\longrightarrow s' \\ c_k=1}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log\left(\gamma_k(s,s')\right)+\log\left(\beta_k(s')\right)\right) \\ &-\max_{\substack{s\longrightarrow s' \\ c_k=0}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log\left(\gamma_k(s,s')\right)+\log\left(\beta_k(s')\right)\right) \end{aligned} \tag{1.67}$$

$$\begin{aligned} \mathrm{L}^{e,out}(c_k) = &\max_{\substack{s\longrightarrow s' \\ c_k=1}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log(\Pr\left\{y_k|p_k^{s\to s'}\right\})+\log\left(\beta_k(s')\right)\right) \\ &-\max_{\substack{s\longrightarrow s' \\ c_k=0}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log(\Pr\left\{y_k|p_k^{s\to s'}\right\})+\log\left(\beta_k(s')\right)\right) \end{aligned} \tag{1.68}$$

Les équations précédentes font référence à un algorithme de décodage pour des turbocodes binaires. Dans le cas du décodage de codes double-binaires, les définitions α_k,β_k,γ_k sont équivalentes. La décision s'effectue sur un ensemble de $card(\mathcal{A})-1$ valeurs de Log-Rapport de Vraisemblance définies par la relation (1.69).

$\forall i \in \mathcal{A}^*$

$$\begin{aligned} \mathrm{L}^{a,i}(c_k) = &\max_{\substack{s\longrightarrow s' \\ c_k=i}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log\left(\gamma_k(s,s')\right)+\log\left(\beta_k(s')\right)\right) \\ &-\max_{\substack{s\longrightarrow s' \\ c_k=0}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log\left(\gamma_k(s,s')\right)+\log\left(\beta_k(s')\right)\right) \end{aligned} \tag{1.69}$$

De même, les informations extrinsèques extraites sont définies par la relation (1.70).

$\forall i \in \mathcal{A}^*$

$$\begin{aligned} \mathrm{L}^{e,out,i}(c_k) = &\max_{\substack{s\longrightarrow s' \\ c_k=i}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log(\Pr\left\{y_k|p_k^{s\to s'}\right\})+\log\left(\beta_k(s')\right)\right) \\ &-\max_{\substack{s\longrightarrow s' \\ c_k=0}}{}^*\left(\log\left(\alpha_{k-1}(s)\right)+\log(\Pr\left\{y_k|p_k^{s\to s'}\right\})+\log\left(\beta_k(s')\right)\right) \end{aligned} \tag{1.70}$$

Dans la pratique, le calcul de la fonction max* engendre une complexité de calcul supplémentaire dû au terme correctif. Dans les fait, cette opération est simplifiée par une simple opération de maximum. Le terme en logarithme jacobien est négligé. Cette approximation est d'autant plus vraie que l'écart entre les valeurs est grand. La FIGURE 1.16 quantifie cette erreur en approchant l'opération max* par une opération max.

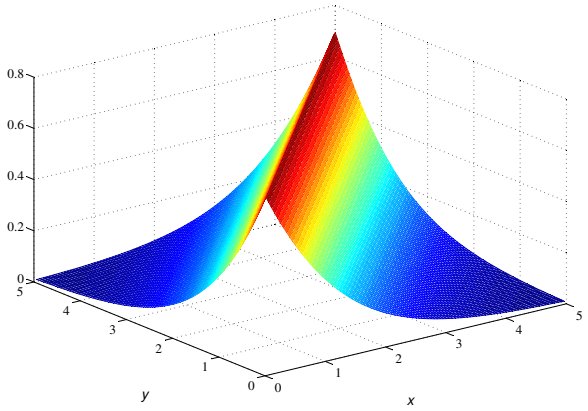


FIGURE 1.16: Approximation de l'écart entre les opérateurs LogMAP et Max-LogMAP pour deux variables

Le décodage avec transformation de treillis de type Radix [56] est exploité sur les codes convolutifs. Cette technique consiste à réunir plusieurs sections de treillis sous contrainte d'obtenir un treillis équivalent avec unicité des chemins de transition. La FIGURE 1.17 montre la transformation d'un treillis binaire à 4 états vers un treillis double-binaire. On parle dans ce cas de transformation de treillis Radix-4. Les sections impaires du treillis disparaissent alors et la taille du treillis est divisée par 2. L'algorithme BCJR n'entraîne pas de dégradation de performance sous réserve d'unicité des chemins de transition. Dans ce cas, les relations (1.59-1.60) deviennent (1.71-1.72).

$$\alpha_{2k+2}(s'') = \sum_{s'} \sum_{s} \alpha_{2k}(s).\left(\gamma_{2k+1}(s,s').\gamma_{2k+2}(s,s'')\right) \tag{1.71}$$

$$\beta_{2k}(s) = \sum_{s'} \sum_{s} \beta_{2k+2}(s'').\gamma_{k+1}(s,s').\left(\gamma_{2k+1}(s,s').\gamma_{2k+2}(s,s'')\right) \tag{1.72}$$

1.5 Architectures de décodage

La section 1.3 fait état de différentes familles de codes correcteurs d'erreurs aux caractéristiques très variées. Parmi ces codes correcteurs, les codes LDPC et les turbocodes convolutifs offrent les meilleurs performances de décodage, mais font

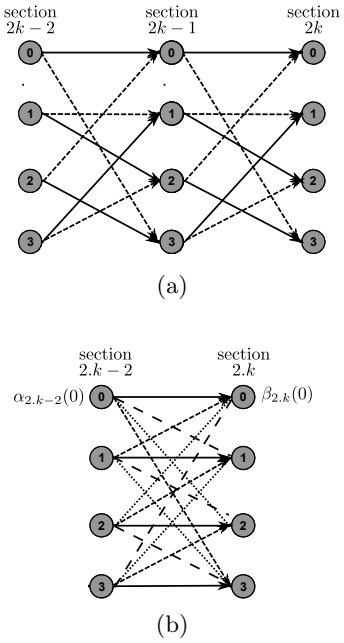


FIGURE 1.17: Transformation d'un treillis binaire (a) en un treillis double-binaire (b)

usuellement appel aux algorithmes de décodage spécifiques détaillés dans la section 1.4. Cette partie propose un état de l'art sur le décodage des codes correcteurs d'erreurs avancés intégrés dans les protocoles standardisés et souligne l'intérêt croissant de structures de décodage génériques et flexibles.

1.5.1 Architectures de décodage dédiées

Cette première partie fournit quelques exemples d'architectures dédiées à un code, à un type de code ou à un standard particulier. Les architectures résultantes font appel à des algorithmes spécifiques permettant de bonnes performances de décodage avec une complexité réduite. Ces architectures sont donc présentées comme point de référence pour des architectures multistandards.

Les architectures dédiées aux codes QC-LDPC sont très présentes dans la littérature. Les standards font appel à de nombreux codes en fonction des choix de Modulation et Schéma de Codage. De ce fait, les architectures de décodage ciblent plusieurs codes ou plusieurs standards comme les codes QC-LDPC des standards IEEE 802.11n et IEEE 802.16m. De nombreuses équipes proposent leurs solutions de décodeurs. [57] présente une architecture de décodage Offset-Min-Sum appliquée aux codes QC-LDPC du standard IEEE 802.11n. Elle a été intégrée sur une cible ASIC 90 nm et permet d'obtenir des débits de 680 Mbps pour une fréquence d'horloge de 346 MHz. [58] propose l'implantation d'un décodeur utilisant l'approximation Λ-3-Min avec un traitement par couches horizontales sur les matrices du standard IEEE 802.16e. L'architecture est intégrée sur une cible ASIC de 130 nm. L'architec-

ture permet d'atteindre 620 Mbps pour une fréquence d'utilisation de 333 MHz. [59] favorise un décodage suivant l'algorithme de Propagation de Croyance avec mise à jour suivant l'approche Somme-Produit. Ce décodeur s'adresse aux standards IEEE 802.11n et IEEE 802.16e avec une implantation sur cible ASIC 130 nm.

D'autres architectures sont proposées sur différents turbocodes. [60] exploite la transformation du treillis suivant la technique Radix-4 appliquée au décodage d'un code binaire (UMTS) avec l'algorithme LogMAP. Il obtient des débits de 25 Mbps sur circuit CMOS de 180 nm avec un seul cœur de processeur. [61] exploite la même technique avec l'algorithme SubMAP sur les mêmes codes. Il obtient des débits de 20 Mbps sur circuit CMOS de 130 nm. L'étude publiée par [62] représente les performances actuellement atteintes par une architecture dédiée au standard 3GPP LTE. Cette architecture bénéficie du parallélisme optimal sur le traitement en fenêtre du code 3GPP LTE. 64 cœurs SISO sont exploités avec un algorithme de calcul MaxLogMAP. Ce décodeur atteint 1,28 Gbps à 400 MHz. En exploitant la technique de Radix-4, [63] dépasse le seuil des 2 Gbits.

L'entreprise TrellisWare [64] commercialise un décodeur Homeplug AV avec des débits de 135 Mbps à 150 MHz. Le décodage conjoint de plusieurs types de codes a été étudié dans la littérature. Pour une architecture de décodage adaptée à deux turbocodes binaire et double-binaire partageant le même nombre d'états, la technique Radix-4 est privilégiée. [65] propose une architecture compatible 3GPP LTE (simple binaire) et 802.16m (double-binaire) avec cette technique et 8 cœurs de processeur en parallèle.

Toutes ces architectures sont résumées dans la Table 1.4. La profusion des cibles d'implantation rend difficile la comparaison entre les différentes architectures. Pour le décodage de codes similaires, il est courant de représenter le rapport entre surface et débit au moyen de la métrique TAR [59] en prenant en compte la technologie cible et le nombre d'itérations. Les données de la Table 1.4 font référence à une adaptation sur une technologie ASIC 65 nm pour 6 itérations de décodage turbocode et 10 itérations de décodage LDPC.

1.5.2 Architectures de décodage multistandards

Des structures de décodage multistandards ont été étudiées dans divers contextes. Les architectures reconfigurables permettent de s'adapter aux structures spécifiques des codes. Le parallélisme dédié dans un cas d'usage est modulé suivant les utilisations afin d'atteindre des débits d'utilisation compétitifs comparés à des architectures dédiées. La reconfiguration de la cible permet de réduire les choix dûs à la multiplicité des codes et permet ainsi de bénéficier de fréquences de l'ordre de 400 à 500 MHz, ce qui favorise les débits de décodage. De plus, la puissance nécessaire à l'utilisation de la cible est optimisée par la spécificité du code. La structure allemande FlexiChap [66] se base sur une architecture ASIP pour décoder de nombreux codes courants associés aux standards 3GPP LTE, IEEE 802.16e et IEEE 802.11n. [67] détaille une autre architecture reconfigurable basée sur la structure en treillis des codes QC-LDPC. Les paramètres de décodage spécifique aux décodages de tur-

Réf.	Standard	Algorithme	Surface (*en nm*)	Fréq. (*en MHz*)	Aire (*en* mm^2)	Débit (*en Mbps*)	TAR [a] (*en Mbps* / mm^2)
[59]	*802.11n 802.16e*	*BP SPA (8 it)*	130	235	3.88	65	107
[57]	*802.11n*	*HLBP OMS (10 it)*	90	346	1,77	680	1086
[58]	*802.16e*	*HLBP L3M (10 it)*	130	333	3,83	620	1300
[60]	*UMTS*	*LogMAP (16 it)*	180	145	14,5	24	100
[63]	*LTE*	*LogMAP (6 it)*	65	450	7,7	2,15	279
[62]	*LTE*	*MaxLogMAP (6 it)*	65	400	8,3	1,28	155
[64]	*HPAV*	non communiqué *(4it)*	130	150	2,3	300	700
[65]	*LTE 802.16e*	*MaxLogMAP (8 it)*	130	250	10,7	186	185

a. Normalisée pour une technologie ASIC TSMC 65 nm, avec 10 itérations (QC-LDPC) et 6 itérations (turbocodes)

TABLE 1.4: Résultats d'implantation d'architectures de décodage dédiés

bocodes (3GPP LTE) et de codes QC-LDPC (IEEE 802.11n) sont résumés dans la Table 1.5.

Cependant, la reconfiguration de ces architectures nécessite un temps de latence entre les différents usages. Pour modifier la configuration de décodage, il est nécessaire d'effacer la cible et de recharger la structure adaptée. Cette opération peut être optimisée par des reconfigurations de parties spécifiques afin de réduire cette latence. Cependant, le décodage aléatoire à la volée n'est pas envisageable, contrairement à l'objectif principal de ces travaux.

Les algorithmes conjoints ont également été abordés. [68] propose ainsi un algorithme commun pour le décodage de codes LDPC et de turbocodes binaires, basé sur le décodage en treillis et l'algorithme LogMAP. Cependant leur étude ne propose pas d'utilisation multistandard et reste donc sur l'application d'un même algorithme pour décoder différents types de codes sans aborder la problématique de la flexibilité du décodeur. Les premières architectures conjointes apparaissent avec [69], qui propose un décodage des codes IEEE 802.11n, IEEE 802.16m et 3GPP LTE avec des performances fortement diminuées par rapport à des décodeurs dédiés, leur contrainte étant alors de réduire le coût énergétique de décodage.

La Table 1.5 résume les caractéristiques des architectures multistandards décrites.

Réf.	**Parallé -lisme**	**Itérations**	**Circuit**	**Fréquence** *en MHz*	**Débits** *en Mbps*
Décodeur LDPC (Wi-Fi)					
[66]	96	10-20	ASIP (65nm)	400	34-257
[67]	96	75	ASIP (65nm)	320	140
[68]	81	15	ASIC (90nm)	500	500
[69]	11	8	ASIC (45nm)	150	122
Décodeur Turbo (LTE)					
[66]		5	ASIP(65nm)	400	18
[67]	24	18	ASIP(65nm)	320	140
[68]	1-12	6	ASIC(90nm)	500	450
[69]	6	8	ASIC(45nm)	150	15,4
[70]	4-8	5,5	ASIC(0,13μm)	302	326

Table 1.5: Comparaison d'implantation d'architectures reconfigurables

1.5.3 Enjeux et perspectives du décodeur multistandard souhaité

L'usage de terminaux de communications multistandards est de plus en plus fréquent, et les équipements doivent s'adapter à cette nouvelle demande. La grande variété des codes correcteurs d'erreurs relève directement des différents usages et caractéristiques de propagation. Pour de nombreuses liaisons filaires, le canal entraîne peu de dégradations, des codes courts et peu complexes suffisent. Cependant,

sur des réseaux bruités comme le canal radio local ou métropolitain, les réseaux sur courants porteurs en ligne ou la communication par satellite, la forte dégradation du signal transmis nécessite l'utilisation des codes correcteurs d'erreurs avancés. Différents types de codes ont été listés dans la section 1.1.1 de ce chapitre. De même, quelques scénarios multistandards ont été exhibés dans la section 1.1.2. Nous avons montré les points communs et les points différents des codes correcteurs avancés et de leur décodage.

Une plateforme de réception multistandard est envisageable dans les scénarios d'usage domestique (boîtiers compatibles Homeplug AV et Wi-Fi pour étendre une couverture domestique) ou mobile (smartphone compatible 3G/4G et Wi-Fi). Or le décodage de canal est une étape primordiale et très consommatrice en ressources matérielles et en énergie. Une architecture de décodage alliant différentes technologies de décodage s'avère de plus en plus commercialement intéressante. La faisabilité de telles structures et leurs coûts d'implantation sont également des données intéressantes pour évaluer la pertinence de telles ou telles interactions.

Le projet de thèse développé dans la suite de ce mémoire a pour but de répondre à une problématique majeure. Est-il envisageable de concevoir une architecture de décodage commune à des codes différents comme les codes QC-LDPC et les turbocodes ? Dans l'affirmative, quelles serait alors les performances de décodage au sens BER ? Quelle serait la latence de décodage entre chaque utilisation ? Quels débits seraient envisageables ? Quelles seraient les limites des codes et des architectures ? Ces questions seront étudiées tout au long de ce mémoire

1.6 Conclusion

Ce chapitre a détaillé la variété des codes correcteurs avancés sélectionnés dans les standards de communication. Sur des canaux de transmissions bruités comme le courant porteur en ligne, les réseaux mobiles ou encore les réseaux sans fils personnels, ceux-ci privilégient des codes correcteurs variés. Un récepteur mobile multistandard doit donc faire face aux caractéristiques de décodage de ces codes, tout en respectant les contraintes de débit et de latence de chacun de ces standards. Mais le choix d'architectures de décodage dédiées entraîne la multiplication de ressources matérielles pour l'adaptation à chaque code.

Ainsi, pour obtenir un récepteur compétitif en termes de performance de décodage, de surface, de débit et de latence, des efforts de mutualisation sont nécessaires. Plusieurs architectures de décodage conjoint ont été envisagées. La structure des codes QC-LDPC permet l'adaptation de structures de décodage à une gamme de codes prédéfinis. La transformation des treillis suivant la technique Radix permet le décodage conjoint de turbocodes binaires et double-binaires. Cependant, le décodage conjoint de codes QC-LDPC et turbocode fait intervenir majoritairement des cibles matérielles reconfigurables ce qui rend l'adaptation entre les réseaux moins flexible. Pourtant, les codes avancés majoritaires partagent des propriétés analogues et la mutualisation des ressources de décodage reste envisageable. Dans le chapitre

qui suit, nous nous efforcerons de caractériser les architectures de décodage des familles de codes QC-LDPC et turbocode pour sélectionner des algorithmes de calcul permettant un fort degré de mutualisation des ressources matérielles.

Chapitre 2

Métriques de complexité des algorithmes de décodage

La littérature fait état de nombreuses stratégies de codage de canal. Les codes correcteurs d'erreurs avancés sont privilégiés pour protéger les transmissions sur canaux bruités. La recherche de codes aux meilleurs caractéristiques se poursuit parallèlement à la recherche d'algorithmes et d'architectures de décodage performants. Face aux contraintes architecturales, à leur complexité d'implantation ou à leur rapidité d'exécution, certains algorithmes de décodage sont écartés des récepteurs numériques.

Ce chapitre est consacré à l'évaluation des métriques de complexité et de latence d'un algorithme de décodage en fonction des paramètres du code correcteur. Dans un premier temps, les opérateurs classiques sont caractérisés au plus bas niveau matériel ce qui permet de détailler la complexité logique et la latence de chaque opération. Cette étude est appliquée comme référentiel pour caractériser la complexité des architectures liées aux algorithmes courants et à leurs variantes, aussi bien pour le décodage de codes matriciels que de codes en treillis. Au cours de ce chapitre, une empreinte de complexité sera présentée en tant que métrique permettant de caractériser un code et son décodeur associé. Cette empreinte constitue une aide à la sélection d'une famille d'algorithmes pour une architecture regroupant plusieurs types de codes.

2.1 Évaluation de la complexité matérielle

Les algorithmes de décodage se décrivent à partir d'opérateurs simples d'ordonnancement séquentiel ou parallèle. Dans cette partie, les opérateurs de calcul sont caractérisés suivant leur complexité logique et leur temps d'exécution. Cette section introduit l'ensemble des opérateurs étudiés dans la suite de ce chapitre.

2.1.1 Métriques d'évaluation des architectures

Une opération sur un message numérique s'évalue suivant des transformations logiques appliquées à la représentation binaire de ce signal. Ces transformations font intervenir deux critères : la **complexité** logique de l'opération à effectuer et le **délai de calcul** nécessaire pour sa réalisation du calcul.

En prenant en compte les critères matériels, la complexité d'une opération est représentée suivant différentes métriques. Le nombre de portes logiques à 1, 2 ou plusieurs entrées permet de définir une première étape de complexité de calcul. Ces portes logiques sont ensuite déclinées en nombre de transistors ou de transferts à travers des **tables de correspondance** (Look-Up Table ou LUT) suivant les technologies matérielles envisagées.

Le délai de calcul s'évalue avec différents critères. Le **délai de propagation** à travers les éléments logiques fournit une première notion du délai de calcul. Ce temps est évalué en fonction des caractéristiques électriques du circuit de transistors permettant l'opération. Il s'évalue comme un délai entre la modification d'un signal d'entrée et sa répercussion sur le signal de sortie. Cette métrique est difficile à appréhender en pratique. Elle dépend des caractéristiques matérielles de la cible d'intégration, mais également des choix de structure établis par les différents outils de développement. Pour un processus synchronisé sur un signal périodique appelé **horloge**, la modification d'une entrée n'est visible que sur une modification de ce signal de synchronisation (en général, le front montant). Pour les éléments de calcul complexes, il est préférable de séquencer les étapes de calculs en fonction des cycles de l'horloge. Dans ce cas, le délai de calcul s'évalue en fonction de la **latence de calcul** qui correspond au nombre de cycles d'horloge nécessaire entre la modification d'un signal d'entrée et sa répercussion sur le signal de sortie. Dans ce cas, le **chemin critique** représente le délai de propagation maximal pour effectuer une opération sur un cycle d'horloge. Ce chemin contraint la fréquence maximale de l'horloge du système. Le délai de calcul d'une opération s'évalue alors en fonction de cette fréquence maximale et en fonction de la latence de calcul évaluée en cycles d'horloge.

Les choix d'intégration des opérations logiques sont relatifs à la cible matérielle sélectionnée. Les cibles FPGA (*Field-Programmable Gate Array*) [71] sont des architectures reconfigurables structurées. Elles sont caractérisées par leur nombre de blocs logiques configurables (CLB) et de bloc mémoires de tailles spécifiées. Les opérations logiques sont effectuées par transfert à travers des tables de correspondance. La complexité totale d'une architecture de décodage s'évalue alors en taux d'occupation de la cible.

L'intégration sur cible ASIC (*Application-Specific Integrated Circuit*) nécessite la création de masques spécifiques à l'architecture développée. Les opérations logiques sont réalisées par des transistors de différentes tailles (typiquement de 180 à 45 nm). Les bancs de mémoires sont intégrés en fonction des besoins de l'architecture. La complexité d'une architecture s'évalue par la surface logique et par la surface des mémoires dédiées. La surface de silicium équivalente est également choisie comme critère de complexité. La taille des transistors et les choix de placement et routage impactent cette surface et modifient le délai de propagation des opérations.

L'évaluation de la complexité d'une architecture de décodage d'un point de vue matériel est spécifique à la technologie choisie. La comparaison de ces données doit donc être relativisée. Dans ce chapitre, nous présentons un critère de comparaison de la complexité matérielle sans considérer la cible matérielle.

2.1.2 Paramétrisation architecturale des opérateurs

2.1.2.1 Opérateurs génériques

Dans un premier temps, on suppose un opérateur $\star$ effectuant une opération sur un seul opérande noté o_1. Cette opération effectue une transformation d'un message sous différentes formes. Sans plus de détail, on note $\chi_\star$ la complexité relative à la transformation du message et $\delta_\star$ le délai de propagation de celui-ci.

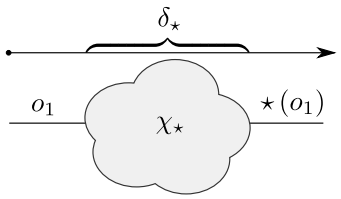


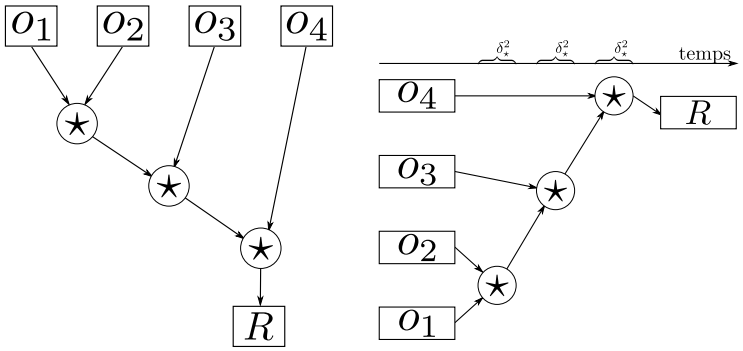
FIGURE 2.1: Représentation de la complexité et du délai de propagation d'un opérateur $\star$

On suppose ensuite que l'opérateur $\star$ effectue une opération sur deux opérandes ou plus. Dans la pratique, ceci revient à effectuer un calcul sur deux éléments consécutifs. L'opérateur $\star$ appliqué à N opérandes o_n nécessite d'exécuter ces opérations dans un ordre précis suivant la distribution (2.1).

$$R = o_1 \star o_2 \star o_3 \star ... \star o_N = (((o_1 \star o_2) \star o_3) \star ...) \star o_N \tag{2.1}$$

Dans ce cas, le délai de propagation relatif au calcul de l'opération (2.1) est noté $\delta_\star^N$. La métrique $\delta_\star^2$ représente le délai de propagation de l'opérateur $\star$ pour deux opérandes. De même, on introduit la complexité de calcul de cette opération par $\chi_\star^N$ et la complexité de l'opération sur deux opérandes par $\chi_\star^2$.

En respectant l'ordre de calcul, l'opération (2.1) est effectuée sur les opérandes successifs, ce qui génère un **arbre binaire** [72] représenté sur la FIGURE 2.2a. La hauteur de l'arbre binaire représente son nombre d'étages. Dans ce contexte, le délai



(a) (b)

FIGURE 2.2: Ordre de calcul pour un opérateur $\star$ à plusieurs variables (a) et son architecture de calcul associé (b)

de propagation de l'opérateur $\star$ est proportionnel à la hauteur de l'arbre. La hauteur de la structure de calcul associée à l'arbre de la FIGURE 2.2a est de $N-1$ étages. De ce fait, le délai de propagation de l'opération correspond à la relation (2.2).

$$\delta_\star^N = (N-1).\delta_\star^2 \tag{2.2}$$

La FIGURE 2.2b représente l'architecture de l'opération (2.1). Cette architecture peut être séquencée en réalisant alors chaque opération $\star$ sur un cycle d'horloge. Dans ce cas, la latence de calcul équivaut à la hauteur de l'arbre et s'établit à $N-1$ cycles d'horloge.

Le nombre de nœuds de l'arbre binaire représente le nombre d'opérateurs $\star$ à deux variables nécessaires au calcul de l'opération (2.1). La complexité du calcul de l'opération est proportionnelle à celle de l'opérateur $\star$ pour deux variables et au nombre de nœuds de son arbre binaire. L'opération (2.1) fait intervenir $N-1$ opérateurs $\star$ à deux variables. De ce fait, la complexité de l'opération s'évalue par la relation (2.3)

$$\chi_\star^N = (N-1).\chi_\star^2 \tag{2.3}$$

Les propriétés de l'opérateur $\star$ permettent de réduire le délai de propagation. Si l'opérateur est commutatif, alors il vérifie la relation (2.4). Si il est associatif, alors il vérifie la relation (2.5).

$\forall a, b, c \in \mathbb{R}$

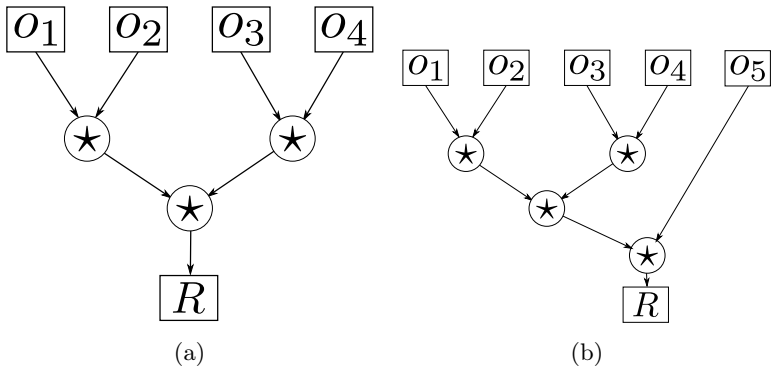
$$a \star b = b \star a \tag{2.4}$$

$$(a \star b) \star c = a \star (b \star c) = a \star b \star c \tag{2.5}$$

Lorsque l'opérateur $\star$ vérifie les propriétés de commutation et d'association, l'ordre des opérations n'influencent plus le résultat. Un enchaînement des opérateurs plus adéquat réduit le délai de propagation sans impacter le résultat de l'opération.

L'enchaînement des opérations suivant l'association (2.6) décrit un autre arbre de calcul pour l'opération (2.1).

$$R = (((o_1 \star o_2) \star (o_3 \star o_4)) ... \star o_N) \tag{2.6}$$



(a) (b)

FIGURE 2.3: Ordre de calcul d'un opérateur commutatif et associatif pour 4 (a) et 5 (b) opérandes

Cet arbre est représenté sur la FIGURE 2.3a lorsque le nombre d'opérandes N correspond à une puissance de 2 et à la FIGURE 2.3b pour un nombre d'opérandes quelconque. Pour ce nouvel ordre de calcul, la complexité $\chi_\star^N$ n'est pas modifiée et respecte la relation (2.3). En effet, la loi d'association ne réduit pas la complexité de calcul de l'opérateur mais l'architecture de celui-ci afin de réduire le délai de propagation. La première étape de calcul fait intervenir $\lfloor \frac{N}{2} \rfloor$ opérateurs $\star$ à deux opérandes, ce qui revient à réduire le nombre d'opérandes à $\lceil \frac{N}{2} \rceil$ à l'étage suivant. En développant ce raisonnement par récurrence, il advient que l'arbre de calcul dispose d'une hauteur de $\lceil \log_2 N \rceil$. Le délai de propagation associé à ce schéma de calcul revient donc à (2.7).

$$\delta_\star^N = \lceil \log_2(N) \rceil . \delta_\star^2 \tag{2.7}$$

Tout comme pour le premier ordonnancement de calcul, cette opération se séquence en $\lceil \log_2 N \rceil$ cycles d'horloge. Le chemin critique de l'opération n'est pas impacté par cet ordre de calcul.

2.1.2.2 Représentation de l'information

Les algorithmes de décodage de canal font principalement intervenir des Log-Rapports de Vraisemblance. Ces informations prennent leurs valeurs dans l'espace des réels $\mathbb{R}$. Pour réduire le délai et la complexité de décodage, ces variables sont représentées sur des ensembles finis d'éléments. La notion de quantification introduite par [73] appliquée à ces informations permet d'établir une plage de valeurs. Afin de simplifier cette représentation, on représente les variables en base 2. On parle de **quantification binaire** [74] [75].

Lorsque ces messages sont des valeurs réelles positives $\mathbb{R}^+$, ils sont représentés sur des ensembles non signés. Ces ensembles sont notés $Qi.f$. Dans ce cas, la partie entière du signal est codée sur i bits et la partie fractionnaire sur f bits. L'ensemble des valeurs est représenté sur $(i+f)$ bits.

Lorsque la valeur codée est représentée dans l'espace des réels $\mathbb{R}$, on utilise une représentation signée. Dans ce cas, $Qi.f$ utilise 1 bit pour coder le signe de l'expression et $i-1$ bits pour représenter la partie entière de la valeur. La partie fractionnaire conserve la dynamique des messages non-signés. De ce fait, cet ensemble est également représenté sur $(i+f)$ bits. L'expression (2.8) résume la quantification d'un entier réel x par son approximation x_q en respectant la symétrie de la représentation par rapport à 0.

$$x_q = \begin{cases} 2^{i+f-1}-1 & si \quad x > 2^{i-1}-2^{-f} \\ -\left(2^{i+f}-1\right) & si \quad x < -\left(2^{i-1}-2^{-f}\right) \\ \lfloor x.2^f + 0.5 \rfloor & sinon \end{cases} \tag{2.8}$$

La FIGURE 2.4 donne un exemple de quantification signée de l'information sur l'ensemble $Q4.2$. L'erreur de quantification ε dans la zone de quantification non saturée vérifie (2.9).

$$\varepsilon < 2^{-f} \tag{2.9}$$

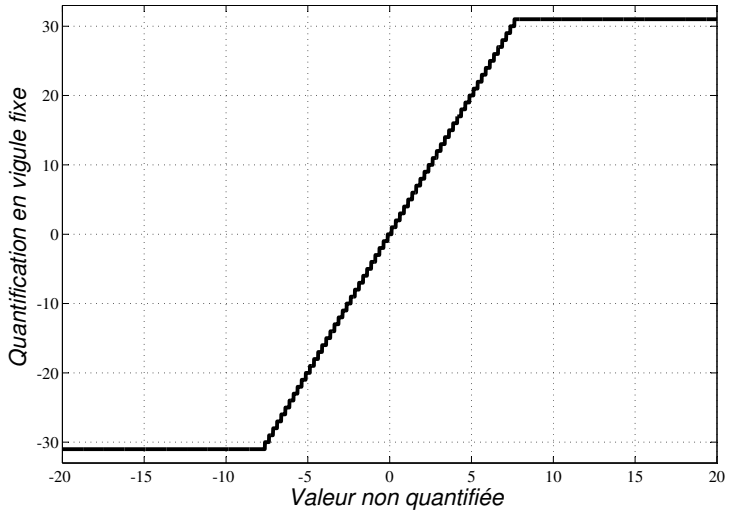


FIGURE 2.4: Quantification d'un message signé suivant la représentation $Q4.2$

Dans la suite, on suppose que les signaux signés sont codés suivant le principe du **complément à deux** [76].

2.1.2.3 Qualification des opérateurs courants

Les algorithmes de décodage courants font intervenir des opérateurs à un ou plusieurs opérandes. Chaque opérateur fait intervenir une complexité en relation

avec la quantification de ces messages. On suppose que chaque message suit la même dynamique et est quantifié sur Q_b bits. Dans cette partie, on propose de qualifier les différents opérateurs qui seront employés dans tout ce chapitre et de dimensionner leur complexité de calcul.

La multiplication (ou la division) suivant des puissances de 2, notée notés $\gg$ (ou $\ll$) se caractérise par un décallage de la dynamique du signal en ajoutant (ou retirant) un bit au message. La complexité (notée $\chi_{\gg}$) et le délai de propagation ($\delta_{\gg}$) de ces opérations sont donc négligeables.

Le changement de signe (noté $-$) d'un message représenté suivant le complément à 2 consiste à inverser tous les signes de ce message. Le résultat est ensuite incrémenté de 1 sur le bit de poids faible. Le délai de propagation de ce message δ_- et sa complexité χ_- sont proportionnels à la quantification choisie Q_b.

La scission S ou la réunification d'un message suivant son signe et son argument représenté suivant le complément à 2 consiste à changer le signe du message si celui-ci est négatif. Ces opérations se déterminent par une opération de changement de signe suivi de la sélection entre deux valeurs Nous notons χ_S la complexité de cette opération et δ_S son délai de propagation. Nous notons également δ_M le délai de propagation dû à la sélection en sortie de l'opération.

Il existe différentes techniques de transformation suivant des opérateurs logarithmiques et exponentiels comme les opérateurs jacobiens ($\delta_{\boxplus}$) ou la transformée en phi (δ_ϕ). [77] a choisi d'intégrer une solution suivant le principe de l'algorithme CORDIC [78] pour le calcul de l'opérateur ϕ. La complexité résultante χ_ϕ et le délai de propagation δ_ϕ s'évaluent en fonctions des fonctions logiques. [79] et [68] privilégient l'enregistrement de ces transformations dans une table de correspondance dont la taille et la profondeur dépendent de la quantification choisie. La complexité de ces opérateurs s'évalue en fonction de la mémoire.

Les algorithmes de calcul de décodage de codes correcteurs font intervenir de nombreux opérateurs à plusieurs variables. On peut citer les additions et soustractions, les multiplications et les Comparaisons/Sélections du minimum et du maximum. Pour comparer chaque opérateur sur un même rapport de complexité, il est important de réduire ces opérations en éléments de calcul à deux opérandes. Pour cela, il est nécessaire de vérifier que ces opérateurs sont associatifs et commutatifs. La Table 2.1 résume les principaux opérateurs utiles pour le décodage de canal. Ces opérateurs vérifient les propriétés de commutation et d'association. De ce fait, les arbres de calcul partagent les propriétés décrites dans la section 2.1.2.1.

Les opérations d'addition et de soustraction de messages représentés suivant le complément à deux sont équivalents. Ces opérations sont effectuées des bits de poids faibles aux bits de poids forts. La complexité (notées χ_+) et le délai de propagation (noté δ_+) sont proportionnels à Q_b. Il en est de même pour les opérations de comparaison (de complexité χ_{CS} et de délai de propagation δ_{CS}) dont les opérations sont effectuées des bits de poids forts aux bits de poids faibles. Une opération de sélection est opérée par un multiplexeur de complexité χ_M.

La multiplication par un scalaire f s'évalue par plusieurs opérations de multiplications ou de divisions par une puissance de 2 suivies d'une série d'additions.

Opérateurs	Notation	Propriété de commutation	Propriété d'association
Addition	$\sum(a,b)$	$a+b=$ $b+a$	$(a+b)+c=$ $a+(b+c)$
XOR	$\oplus(a,b)$	$a\oplus b=$ $b\oplus a$	$(a\oplus b)\oplus c=$ $a\oplus(b\oplus c)$
Multiplication	$\prod(a,b)$	$a\times b=$ $b\times a$	$(a\times b)\times c=$ $a\times(b\times c)$
minimum	$\min(a,b)$	$\min(a,b)=$ $\min(b,a)$	$\min(\min(a,b),c)=$ $\min(a,\min(b,c))$
maximum	$\max(a,b)$	$\max(a,b)=$ $\max(b,a)$	$\max(\max(a,b),c)=$ $\max(a,\max(b,c))$
Logarithme Jacobien	$\boxplus(a,b)$	$\log(e^a+e^b)=$ $\log(e^b+e^a)$	$\log(\exp(\log(e^a+e^b))+e^c)=$ $\log(e^a+\exp(\log(e^b+e^c)))$

TABLE 2.1: Propriétés mathématiques des principaux opérateurs associés aux algorithmes de décodage

La complexité $\chi_{\times f}$ et le délai de propagation $\delta_{\times f}$ sont de l'ordre d'une opération d'addition sur plusieurs variables dont le nombre dépend du poids du scalaire f (noté q_f). Ainsi, chaque cas de multiplication doit être étudié pour en connaître ses métriques. La complexité relative est proportionnelle à $Q_b \times q_f$. Le délai de calcul suit la relation $\log_2(q_f).\delta_+^2$

La multiplication de deux éléments est effectuée par une suite de Q_b-1 additions de signaux quantifiés de Q_b à $2.Q_b-1$ bits. La complexité de calcul est de l'ordre de Q_b^2 et le délai de propagation de l'ordre de $\lceil\log_2(Q_b)\rceil.\delta_+^2$.

2.1.3 Comparaisons de codes et protocoles expérimentaux

Dans le chapitre 1, il est établi que les codes sont caractérisés par les triplets d'informations (N, K, d_{min}). Cependant, connaître la distance minimale associée à des codes correcteurs d'erreur avancés comme les turbocodes [46] et les codes LDPC [80] est complexe. Dans la pratique, ces codes font intervenir des algorithmes de décodage itératifs dont le nombre d'itérations impacte à la fois les performances BER et FER de décodage, mais aussi la complexité globale de décodage et la latence de calcul. Il existe donc de multiples angles de comparaison de la complexité de décodage. Ce chapitre vise à évaluer la complexité relative à un algorithme de décodage dans le but de sélectionner un algorithme de décodage conjoint au décodage de codes LDPC et de turbocodes afin de mutualiser au mieux les ressources matérielles (mémoires et logiques). À partir de cet algorithme, chaque choix de paramètres doit être étudié sur un rapport de complexité de décodage et de gain de performance.

Pour ce faire, on introduit une représentation sur un diagramme en **toile d'araignée** représentant les différents opérateurs cités dans la section 2.1 : les multipli-

cations de messages, les multiplications par un scalaire fixé, les additions et soustractions, les opérations de changement de signe et les comparaisons/sélections de donnée. Les opérateurs les moins complexes ne sont pas représentés. Les opérateurs plus complexes comme l'opération $\max^*$ et la transformée en ϕ sont externalisés et représentés par des transformations à travers une table de correspondance stockée dans une mémoire. Suivant les détails de comparaison le diagramme en toile d'araignée représente ces opérateurs suivant leurs nombre littéral ou en fonction du nombre de portes logiques et d'emplacements mémoires nécessaires au décodage étudié. Les mémoires internes sont dédiées aux calculs des mises à jour des variables (décodage suivant une matrice de parité) et aux données requises au décodage des codes en treillis. Les mémoires externes font référence aux données stockées par le processus itératif et relatives aux informations intrinsèques et aux informations extrinsèques.

La FIGURE 2.5 illustre une représentation possible de l'ensemble de la complexité associée au décodage de tout type de code et suivant tout type d'algorithme de décodage. Cet exemple est donné en fonction des calculs littéraux. La représentation permet d'établir une empreinte caractéristique d'un code et d'un décodeur associé.

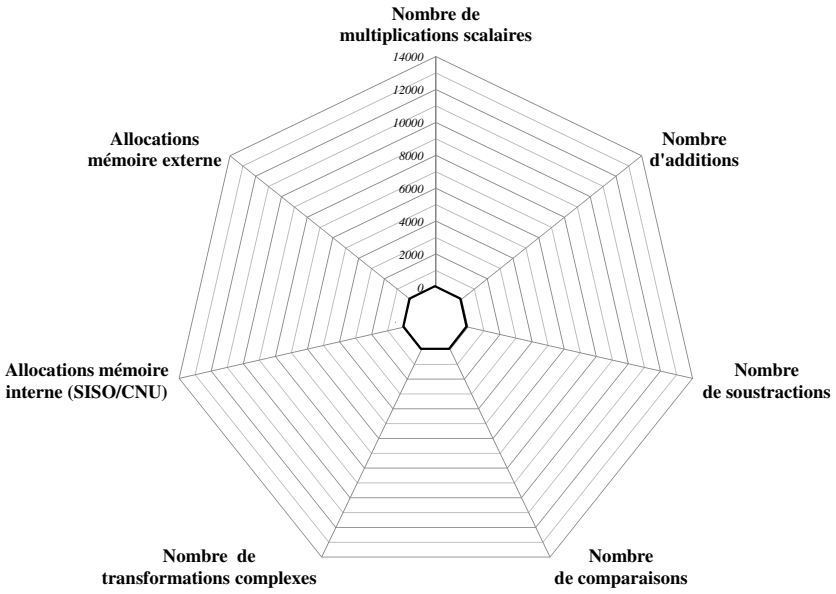


FIGURE 2.5: Représentation des opérateurs courants sur un diagramme en toile d'araignée

La représentation sur un diagramme en toile d'araignée présente de multiples intérêts. Il permet en premier lieu de caractériser la complexité d'un algorithme en fonction des opérateurs qu'il utilise. Dans la pratique, les cibles matérielles de décodage font appel à des processus de calcul différents. Les FPGA incluent des

multiplieurs ce qui permet d'externaliser la complexité des multiplications sur deux messages. Le choix d'intégration des opérateurs étant spécifique à la technologie matérielle ciblé, il est difficile de comparer chaque opérateur sur un critère universel. L'empreinte d'un décodeur permet de comparer un code et son algorithme de décodage. En normalisant la représentation suivant une métrique préétablie (nombre d'informations systématiques K, nombre de nœuds de la matrice de parité $\mathbf{d}^H$, nombre de section de treillis équivalent par itération de décodage T_{It}, nombre d'itérations nécessaires, performance BER ou FER équivalent, ...), cette représentation permet de comparer différentes stratégies de codage et de décodage de canal.

La suite de ce chapitre propose d'étudier le rapport complexité/performance pour effectuer une comparaison des stratégies de décodage suivant différents critères et sélectionner une piste d'algorithme pour une architecture de décodage multistandard.

2.2 Architectures de décodage matriciels

Les codes LDPC sont définis à partir de conditions de parité entre les variables. Ces conditions sont représentées par une matrice de parité H. Les algorithmes de propagation de croyance (BP) sont adaptés au décodage de ces codes. Dans cette section, les architectures suivant ces algorithmes et leur complexité sont comparées en fonction des caractéristiques de la matrice de parité et au moyen des métriques présentées dans la section 2.1.

2.2.1 Architectures de mises à jour

Une représentation de Tanner permet d'observer les relations entre plusieurs variables suivant une équation de parité. Le chapitre 1 référence de nombreux algorithmes de mise à jour des nœuds de parité. Nous ne reprendrons ici que les mises à jour *Sum-Product* (1.46), *Min-Sum* (1.48) et *Normalized Min-Sum* (1.50). Dans cette partie, on cherche à représenter le coût architectural de la mise à jour d'une variable ou d'un ensemble de variables associées à une condition de parité. Cette architecture s'applique donc aux décodages de codes représentés par une matrice de parité. Nous fixons ici une équation de parité e_m de degré de parité d_c^m. Les indices de cette équation sont repérés dans l'ensemble $\mathcal{N}_m$. En reprenant les notations introduites dans la partie 1.2.3, $\mathrm{L}^a(c_n)$ représente l'ensemble de l'information connue sur la variable c_n et $\mathrm{L}^e(c_n)$ représente l'information extraite du traitement de la mise à jour.

2.2.1.1 Mise à jour Sum-Product

La mise à jour SPA respecte la formule (1.46) introduite dans le chapitre 1. Cette opération se scinde en une décision sur le signe de l'opération et une opération sur la confiance attribuée à cette décision. L'introduction de la fonction ϕ (1.43) permet

le calcul de cette confiance. L'opération (1.46) se résume au système d'équations (2.10).

$$\begin{cases} \operatorname{sgn}\left(\mathrm{L}^{e}\left(c_{n}\right)\right) = \bigoplus\limits_{n' \in \mathcal{N}_{m}, n' \neq n} c_{n'} \\ \left|\mathrm{L}^{e}\left(c_{n}\right)\right| = \phi^{-1}\left(\sum\limits_{n \in \mathcal{N}_{m|n}} \phi\left(\left|\mathrm{L}^{a}\left(c_{n'}\right)\right|\right)\right) \end{cases} \tag{2.10}$$

Les opérations sont séquencées, ce qui mène à une architecture de décodage décrite par l'algorithme de calcul lié à cette opération. Cet algorithme est décrit sur la FIGURE 2.6a. Par convention, les messages non signés sont représentés sur des traits pleins, les messages signés sont en traits mixtes et les messages binaires en traits pointillés.

1. Les informations connues $\mathrm{L}^{a}\left(c_{n'}\right)$ sont recherchées aux emplacements mémoire dédiés. Elles sont indicées par l'ensemble $\mathcal{N}_{m|n}$. La complexité de cette étape consiste à effectuer des opérations de lecture de mémoire sur $(d_c - 1)$ emplacements.

2. Ces messages sont scindés suivant leur signe et leur argument, qui impose un délai de propagation δ_s et une complexité χ_s par variable.

3a. Chaque argument est transformé suivant l'opérateur ϕ. Cette opération est intégralement parallélisée et impose un délai de propagation de δ_{ϕ}. Les $d_c^m - 1$ transformées en ϕ sont ensuite sommées. En reprenant les notions précédemment établies, cette étape nécessite un délai de propagation de $\delta_{+}^{d_c^m - 1}$. Le résultat de la somme subit la transformée inverse de ϕ.

3b. En parallèle à l'étape **3a.**, Une opération XOR est appliquée sur le signe des informations intrinsèques.

4. L'information de sortie $\mathrm{L}^{e}\left(c_{n}\right)$ prend en compte les résultats obtenus des étapes **3a.** et **3b.**. Le résultat est enregistré à l'emplacement mémoire dédié pour une utilisation ultérieure.

Ce processus de calcul est repris sur la FIGURE 2.6a. Cette architecture est parallélisée pour en réduire le délai de propagation Δ_{SPA}^{1}, qui est estimé par la relation (2.11).

$$\Delta_{SPA}^{1} = \delta_S + \max\left(\delta_{\phi} + \delta_{+}^{d_c^m - 1} + \delta_{\phi^{-1}}, \delta_{\oplus}^{d_c^m - 1}\right) + \delta_S \tag{2.11}$$

Cette architecture ne met à jour qu'une variable. L'ensemble des variables d'une équation de parité est mise à jour successivement afin d'éviter les conflits d'accès. Il advient que le délai de propagation nécessaire à la mise à jour d'un nœud de parité est de $d_c^m.\Delta_{SPA}^{1}$.

Les étapes de calcul des mises à jour des variables d'une équation de parité e_m peuvent être parallélisées afin de réduire le délai de calcul. Pour cela, on introduit les probabilités de confiance de l'équation Φ_m et le signe de l'opération Θ_m. Le système (2.10) est équivalent au système (2.12).

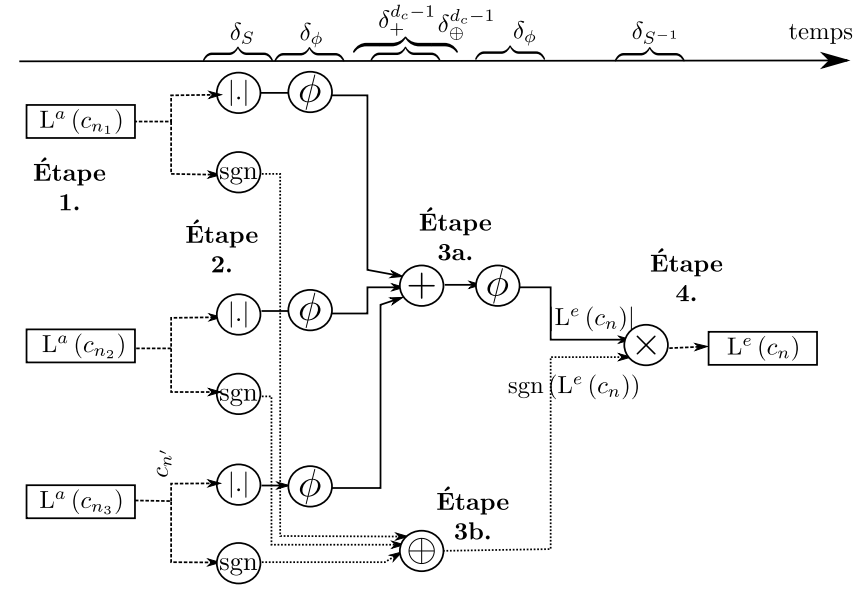


FIGURE 2.6: Architecture de mise à jour SPA pour une variable c_n (a) et architecture optimisée pour une équation e_m (b)

$$\begin{cases} \operatorname{sgn}\left(\mathrm{L}^{e}\left(c_{n}\right)\right) = c_{n} \oplus \underbrace{\bigoplus_{n' \in \mathcal{N}_{m}} c_{n'}}_{\Theta_{m}} \\ \left|\mathrm{L}^{e}\left(c_{n}\right)\right| = \phi^{-1}\left(\underbrace{\sum_{n' \in \mathcal{N}_{m}} \phi\left(\left|\mathrm{L}^{a}\left(c_{n'}\right)\right|\right)}_{\Phi_{m}} - \left(\left|\mathrm{L}^{a}\left(c_{n}\right)\right|\right)\right) \end{cases} \tag{2.12}$$

Les deux éléments Θ_m et Φ_m sont communs aux mises à jour de toutes les variables de l'équation e_m. L'architecture résultante est montrée sur la FIGURE 2.6b. Ceci permet de réduire la complexité du calcul de la mise à jour de toutes les variables de l'équation mais engendre une étape de calcul supplémentaire qui se traduit sur le délai de propagation de cette mise à jour par (2.13).

$$\Delta_{SPA}^{2} = \delta_{S} + \max\left(\delta_{\phi} + \delta_{+}^{d_{c}^{m}} + \delta_{+}^{2} + \delta_{\phi^{-1}}, \delta_{\oplus}^{d_{c}^{m}} + \delta_{\oplus}^{2}\right)\delta_{S} \tag{2.13}$$

La TABLE 2.2 évalue la complexité littérale des deux architectures de mises à jour présentées. Cette complexité est normalisée pour la mise à jour d'une équation de parité e_m. Le délai de propagation de calcul d'une architecture parallélisée par équation de parité vérifie alors Δ_{SPA}^{2}, tandis que celui d'une architecture basique vérifie $d_c.\Delta_{SPA}^{1}$. Cette architecture de mise à jour a été étudiée par [81].

Opérateurs	**Architecture basique**	**Architecture parallélisée par équation de parité**
XORs	$d_c^m.(d_c^m - 2)$	$2.(d_c^m - 1)$
Inversions de signe	d_c^m	d_c^m
Additions	$d_c^m.(d_c^m - 2)$	$2.(d_c^m - 1)$
Transformées en ϕ/ϕ^{-1}	$d_c^m.d_c^m + d_c^m$	$2.d_c^m$

TABLE 2.2: Complexité littérale de la mise à jour des variables associées à un nœud de parité pour l'algorithme SPA

D'autres architectures de mise à jour des nœuds de variables sont également étudiées dans la littérature. [81] propose la mise à jour de ces nœuds en parcourant les variables dans le sens montant et le sens descendant des variables à l'instar du parcours d'un treillis.

2.2.1.2 Mise à jour MS et NMS

La mise à jour MS respecte la formule (1.48) établie dans le chapitre 1. Tout comme pour la mise à jour SPA, cette opération est scindée en une décision dure sur

le signe de l'opération et une opération sur une confiance attribuée à cette décision. L'opération (1.48) se résume au système d'équations (2.14).

$$\begin{cases} \operatorname{sgn}\left(\mathrm{L}^{e}\left(c_{n}\right)\right) = \bigoplus_{n' \in \mathcal{N}_{m}, n' \neq n} c_{n'} \\ \left|\mathrm{L}^{e}\left(c_{n}\right)\right| = \min_{n' \in \mathcal{N}_{m|n}} \left(\left|\mathrm{L}^{a}\left(c_{n'}\right)\right|\right) \end{cases} \tag{2.14}$$

Cette mise à jour simplifie l'architecture de calcul de mise à jour. Les transformées en ϕ sont supprimées et l'étape d'addition est remplacée par une étape de comparaison/sélection de complexité proche. Le délai de calcul de l'opération en est donc également réduit. Pour une mise à jour normalisée (NMS), une étape de multiplication par le scalaire f_{Λ} est ajoutée. Cette nouvelle architecture est présentée sur la FIGURE 2.7a. Dans ce cas, le délai de propagation respecte (2.15).

$$\Delta_{NMS}^{1} = \delta_{S} + \max(\delta_{\min}^{d_{c}^{m}-1}, \delta_{\oplus}^{d_{c}^{m}-1}) + \delta_{S} + \delta_{\times f} \tag{2.15}$$

Tout comme pour l'algorithme SPA, les algorithmes de calcul MS et NMS se parallélisent. En effet, la valeur minimale des ensembles $\mathcal{N}_{m|n}$ est commune à tous les indices n sauf pour un ensemble $\mathcal{N}_{m|n'}$. Il est donc possible d'effectuer une opération de tri afin de ne conserver que les deux confiances les moins fiables des ensembles de parité. Cette architecture de calcul nécessite alors d'effectuer deux comparaison/sélections par opérandes et de garder en mémoire les deux éléments les moins fiables notés ici $\min_1$ et $\min_2$. Cette opération nécessite d'effectuer $2.N-3$ Comparaisons/Sélections pour une complexité de $(2.N-3).\chi_{CS}^{2}$ avec un délai de calcul de $N.\delta_{CS}^{2}$. La FIGURE 2.8 reprend cette architecture de calcul.

Dans ce cas, le délai de calcul est représenté par la relation (2.16).

$$\Delta_{NMS}^{2} = \delta_{S} + \max(d_{c}^{m}.\delta_{CS}^{2} + \delta_{\min}^{2}, \delta_{\oplus}^{d_{c}^{m}-1}) + \delta_{S} + \delta_{\times f} \tag{2.16}$$

Les coûts littéraux de ces deux architectures sont indiqués dans la TABLE 2.3. La structure de décodage parallélisée reste moins complexe malgré l'opération de tri que la structure de mise à jour classique. Cette architecture est notamment reprise par [82]. L'index de la confiance minimale est cependant conservé afin d'éviter l'étape de sélection présenté ici.

2.2.2 Architectures d'ordonnancement par propagation de croyances

2.2.2.1 Ordonnancement par inondation

Les algorithmes de propagation de croyance mettent en relief les relations entre les équations de parité associées à une matrice de parité. L'ordonnancement par inondation correspond à l'ordre de calcul défini par R.G. Gallager [11] dès l'invention des codes LDPC.

La mise à jour des nœuds de variable renvoie à l'ordonnancement associé. L'ordonnancement par inondation respecte les équations (1.54) et (1.55) énoncées dans

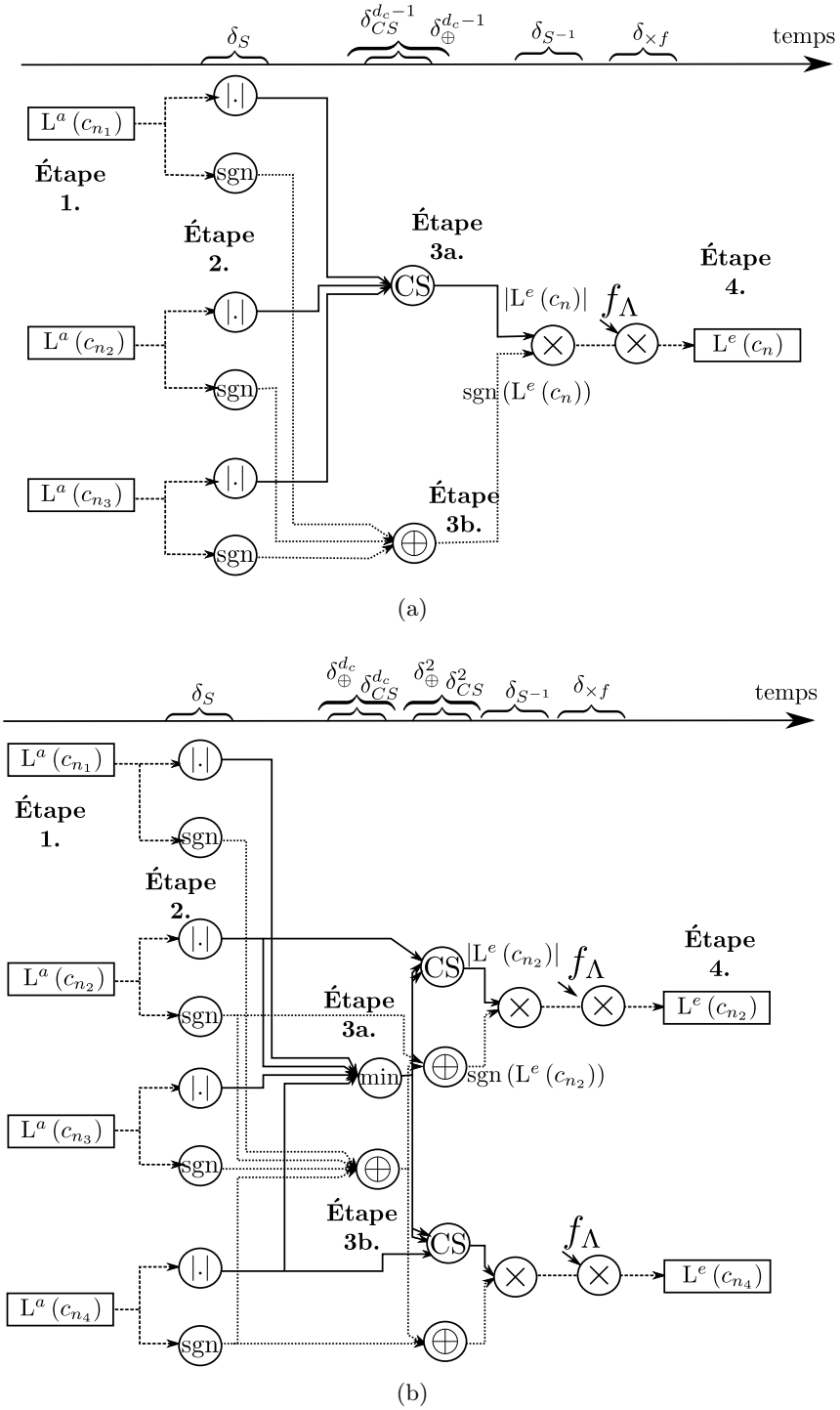


FIGURE 2.7: Architecture de mise à jour NMS pour une variable c_n (a) et architecture optimisée pour une équation e_m (b)

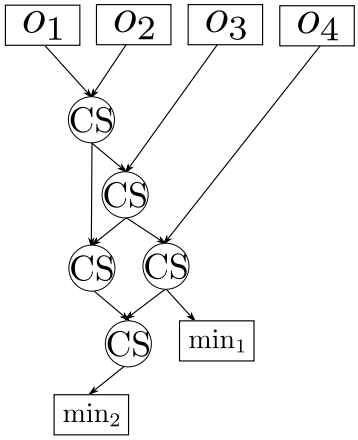


FIGURE 2.8: Arbre de tri des deux valeurs minimales utilisé pour le calcul MS et NMS

Opérateurs	**Architecture basique**	**Architecture parallélisée par équation de parité**
XORs	$d_c^m.(d_c^m - 2)$	$2.(d_c^m - 1)$
Inversions de signe	d_c^m	d_c^m
Comparaisons/Sélections	$d_c^m.(d_c^m - 2)$	$3.d_c^m - 3$
Multiplications scalaires	0 (MS) / d_c^m (NMS)	0 (MS) / d_c^m (NMS)

TABLE 2.3: Nombre d'opérations nécessaires à la mise à jour des variables associées à un nœud de parité pour les algorithmes MS et NMS

le chapitre 1. Afin d'éviter toute auto-confirmation dans le passage de message, l'information fournie à chaque équation de parité respecte la relation (2.17)

$$\mathrm{L}^{a}_{m}\left(c_n\right) = \mathrm{L}^{c}\left(c_n\right) + \sum_{m' \in \mathcal{M}_{n|m}} \mathrm{L}^{e}_{m'}\left(c_n\right) \tag{2.17}$$

La mise à jour du nœud de variable s'effectue en deux étapes. Tout d'abord, l'information *a posteriori* est décidée à travers la remontée des informations $\mathrm{L}^{e}_{m}\left(c_n\right)$ de chaque nœud de parité e_m relativement à une variable c_n. Cette information $\mathrm{L}^{e}_{m}\left(c_n\right)$ est conservée et retranchée de l'information *a posteriori* avant d'être réutilisée par la structure de calcul de mise à jour de parité. De ce fait, l'équation (2.17) équivaut à la relation (2.18).

$$\mathrm{L}^{a}_{m}\left(c_n\right) = \mathrm{L}^{a}\left(c_n\right) - \mathrm{L}^{e}_{m}\left(c_n\right) \tag{2.18}$$

Les informations extrinsèques $\mathrm{L}^{e}_{m}\left(c_n\right)$ provenant des équations de parité sont stockées ainsi que l'information *a posteriori* $\mathrm{L}^{a}\left(c_n\right)$. $d^{n}_{v}+1$ Log-Rapports de Vraisemblance sont stockées par variable c_n. L'information *a posteriori* nécessite le calcul de d^{n}_{v} opérandes et la remontée des informations au niveau des nœuds de contraintes engendre d^{n}_{v} soustractions de deux opérandes pour chaque variable.

Le traitement de l'ordonnancement par inondation vérifie donc la complexité donnée par la Table 2.4 pour chaque nœud de variable.

Opérateurs	**Architecture basique**	**Architecture parallélisée par nœud de variable**
Inversions de signe	0	d^{n}_{v}
Additions	$d^{n}_{v}.(d^{n}_{v}-1)$	$2.d^{\mathbf{n}}_{v}$
Données stockées	$d^{n}_{v}+1$	$d^{n}_{v}+1$
Délai de propagation	$d^{n}_{v}.\delta_{+}^{d^{n}_{v}-1}$	$\delta_{+}^{d^{n}_{v}}+\delta_{INV}+\delta_{+}^{2}$

Table 2.4: Nombre d'opérations nécessaires à la mise à jour d'un nœud de variable c_n pour un algorithme par inondation

2.2.2.2 Ordonnancement par couches horizontales

L'ordonnancement par couches horizontales propose d'effectuer la mise à jour de l'information de décision après chaque traitement de nœud de contrainte. Avant chaque nœud de contrainte e_m, l'information extrinsèque de cette équation est retranchée, respectant la relation (2.19). L'information *a posteriori* sur la variable c_n est mise à jour suivant (2.20).

$$\mathrm{L}^{ap}_{m}\left(c_n\right) = \mathrm{L}^{ap}\left(c_n\right) - \mathrm{L}^{e,old}_{m}\left(c_n\right) \tag{2.19}$$

$$\mathrm{L}_m^a(c_n) = \mathrm{L}^{ap}(c_n) + \mathrm{L}_m^{e,new}(c_n) \tag{2.20}$$

Le traitement par couches horizontales entraîne la complexité décrite par la TABLE 2.5. $d_v^n + 1$ données doivent être stockées lors d'une itération de décodage. Le délai de propagation ne prend pas en compte le parallélisme de calcul potentiel.

Opérateurs	**Algorithme par couche**
Inversions de signe	d_v^n
Additions	$2.d_v^n$
Délai de propagation	$2.d_v^n.\delta_+^2$
données stockées	$d_v^n + 1$

TABLE 2.5: Nombre d'opérations par itération sur la mise à jour d'une variable c_n suivant l'ordonnancement par couches horizontales

2.2.3 Architectures de décodage

Les premières sections font référence aux coûts des processus de mises à jour des nœuds de variables et des nœuds de parité. Cette partie évalue la complexité par itération des choix entre les différents algorithmes. Pour cela, on introduit $\mathbf{d}^H$ comme le nombre d'éléments non-nuls associés à la matrice H. D'après les notations précédemment établies, cette relation vérifie la relation (2.21).

$$\mathbf{d}^H = \sum_{n=1}^{N} d_v^n = \sum_{m=1}^{M} d_c^m \tag{2.21}$$

L'algorithme de décodage par inondation permet d'appliquer les parallélismes de calcul des mises à jour des nœuds de parité et de nœuds de variable. La complexité globale s'en trouve donc améliorée. La TABLE 2.6 reprend ces parallélismes pour évaluer la complexité logique et les délais de calcul nécessaires pour une itération de décodage. Ces calculs sont normalisés par rapport au nombre de nœuds de la matrice de parité $\mathbf{d}^H$.

La TABLE 2.6 résume également la complexité totale de décodage par itération en fonction des algorithmes de mises à jour des nœuds de parité présentés dans la section 2.2.1 et des choix d'ordonnancement de mise à jour des nœuds de variable proposés dans la section 2.2.2. Ce tableau représente la complexité littérale en fonction des caractéristiques des matrices de parité. Cette complexité s'exprime en fonction des nœuds de la matrice de parité $\mathbf{d}^H$.

2.2.4 Choix algorithmiques et compromis complexité/performance

Les architectures décrites dans la section 2.2 montrent un large choix de décodage d'un code matriciel en fonction des algorithmes de propagation de croyances. Nous

Opérateurs	Sum-Product	Min-Sum	Normalized Min-Sum
Architectures par inondation			
XORs	$2.\mathbf{d}^H - 2.M$	$2.\mathbf{d}^H - 2.M$	$2.\mathbf{d}^H - 2.M$
Inversions de signe	$\mathbf{d}^H + \mathbf{d}^H$	$\mathbf{d}^H + \mathbf{d}^H$	$\mathbf{d}^H + \mathbf{d}^H$
Additions	$4.\mathbf{d}^H - 2.M$	$2.\mathbf{d}^H$	$2.\mathbf{d}^H$
Comparaisons/Sélections	0	$2.\mathbf{d}^H - M$	$2.\mathbf{d}^H - M$
Multiplications scalaires	0	0	$\mathbf{d}^H$
Transformées en ϕ/ϕ^{-1}	$2.\mathbf{d}^H$	0	0
Données stockées	$\mathbf{d}^H + N$	$\mathbf{d}^H + N$	$\mathbf{d}^H + N$
Architectures par couches horizontales			
XORs	$2.\mathbf{d}^H - 2.M$	$2.\mathbf{d}^H - 2.M$	$2.\mathbf{d}^H - 2.M$
Inversions de signe	$\mathbf{d}^H + \mathbf{d}^H$	$\mathbf{d}^H + \mathbf{d}^H$	$\mathbf{d}^H + \mathbf{d}^H$
Additions	$4.\mathbf{d}^H - 2.M$	$2.\mathbf{d}^H$	$2.\mathbf{d}^H$
Comparaisons/Sélections	0	$2.\mathbf{d}^H - M$	$2.\mathbf{d}^H - M$
Multiplications scalaires	0	0	$\mathbf{d}^H$
Transformées en ϕ/ϕ^{-1}	$2.\mathbf{d}^H$	0	0
Données stockées	$\mathbf{d}^H + N$	$\mathbf{d}^H + N$	$\mathbf{d}^H + N$

TABLE 2.6: Nombre d'opérations nécessaires à la mise à jour de tous les nœuds par itération

évaluons ici la performance de ces architectures de décodage associées à certains codes QC-LDPC afin de proposer des pistes architecturales pertinentes dans le cadre d'un décodage conjoint.

2.2.4.1 Ordonnancement des calculs pour décodage de matrice de parité

Le décodage de codes QC-LDPC fait intervenir différentes stratégies de décodage. L'ordonnancement des mises à jour suivant l'algorithme de propagation de croyance par inondation correspond à l'ordonnancement défini par R.G. Gallager dans sa thèse [11]. [52] fait référence à un ordonnancement par couches horizontales. Ces deux types d'ordonnancement ont été évalués en fonction de leur complexité arithmétique de calcul. La matrice IEEE 802.16m de taille $N = 2304$ et de rendement $R = 1/2$ a été sélectionnée pour évaluer ces deux types d'ordonnancements. La mise à jour SPA est optimale pour le décodage d'équations de parité. Pour examiner l'impact de la complexité de ces deux algorithmes, le nombre d'itérations a été fixé afin d'atteindre un taux d'erreur binaire de 10^{-5} pour un SNR de 2 dB.

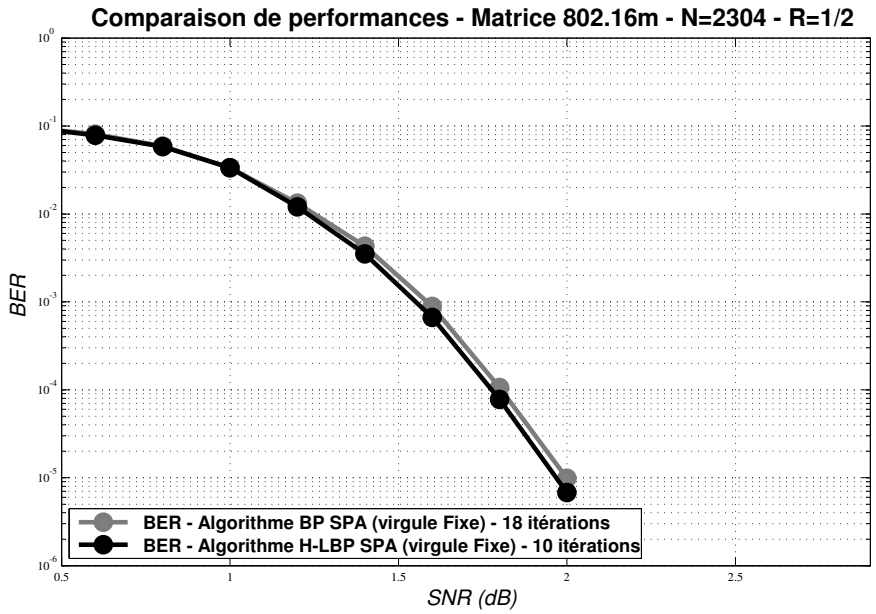
La performance de ce choix est représenté sur la Figure 2.9a. Le seuil de performance recherché est de 18 itérations pour l'ordonnancement par inondation et 10 itérations pour l'ordonnancement par couches horizontales. L'ordonnancement par couches horizontales converge donc deux fois plus rapidement que l'ordonnancement par inondation, ce que confirme [83]. Le diagramme en toile d'araignée introduit dans la section 2.1.3 est utilisé ici pour comparer l'impact de l'ordonnancement sur la complexité totale de l'algorithme à performance équivalente. La Figure 2.9b compare ces complexités architecturales. La rapidité de convergence de l'ordonnancement par couches horizontales compense le surcoût engendré par l'algorithme en couches horizontales et réduit le nombre de requêtes mémoires. La mise à jour par couches horizontales permet de préserver les ressources matérielles à performance constante.

2.2.4.2 Choix de mise à jour de parité

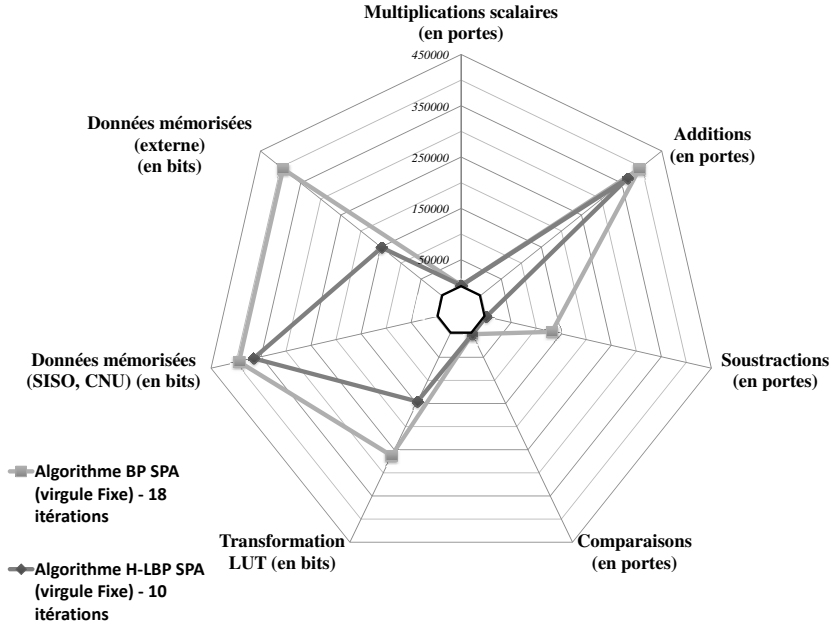
La section 2.2.1 fait état de différents algorithmes de mise à jour. Dans cette partie, le choix entre ces algorithmes est étudié uniquement en termes de performances. L'architecture de décodage par couches horizontales a été privilégiée pour cette comparaison. Différents paramètres ont été testés pour la mise à jour NMS. Le code QC-LDPC sélectionné correspond à la matrice IEEE 802.11n de taille $N = 648$ et de rendement $R = 1/2$. La Figure 2.10 résume les performances de décodage entre les mises à jour SPA, MS et différents paramètres de mise à jour NMS.

Les performances de décodage sont grandement améliorées par le choix d'une mise à jour suivant le NMS. Le facteur d'échelonnage de l'algorithme NMS 0.80 est le plus prometteur sur la Figure 2.10. Cependant, ce choix modifie la complexité de calcul. Il engendre une approximation et un nombre d'additions équivalent qui varient en fonction de la quantification du signal. La Table 2.7 résume ces approches.

La Table 2.7 montre l'impact de la quantification des messages sur la complexité de décodage. Pour tester le choix de facteur 0.80, l'algorithme H-LBP avec la mise



(a)



(b)

FIGURE 2.9: Complexité (b) de décodage à performance comparable (a) pour les algorithmes de propagation de croyance par inondation et par couches horizontales

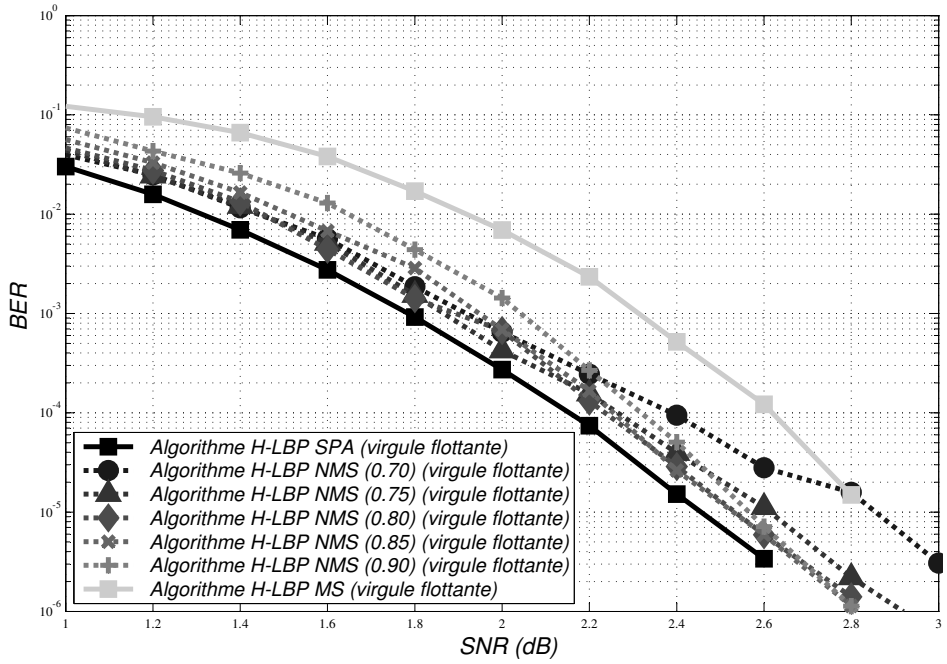


FIGURE 2.10: Performances de décodage de plusieurs algorithmes de mise à jour appliqués à la matrice IEEE 802.11n $N = 648$ $R = 1/2$ pour un ordonnancement par couches horizontales avec 30 itérations

Quantification	**Représentation binaire**	**Approximation décimale**	**Nombre d'additions associées à la multiplication**
$Qx.1$	1	0.5	0
$Qx.2$	3	0.75	1
$Qx.3$	6	0.75	1
$Qx.4$	13	0.8125	2
$Qx.5$	26	0.8125	2
$Qx.6$	51	0.769	3
$Qx.7$	102	0.769	3
$Qx.8$	205	0.8008	4

TABLE 2.7: Complexité d'échelonnage en fonction de la quantification et du facteur sélectionné

à jour NMS a été testé sur une architecture de décodage à virgule fixe avec une quantification de l'information du canal sur $Q4.1$ et sur l'information interne de $Q5.6$. De ce fait, les approximations 0.75 et 0.8125 ont été testées. Pour cela, le nombre d'itérations a été fixé afin d'atteindre un BER de 10^{-5} pour un SNR de 2 dB. Dans ce cas d'usage, le facteur 0.8125 converge une itération plus tôt sur la matrice QC-LDPC du standard IEEE 802.16m pour $N = 2304$ et $R = 1/2$. La performance équivalente est montrée sur la FIGURE 2.11. Ces simulations montrent que l'algorithme avec le facteur 0.8125 converge plus rapidement.

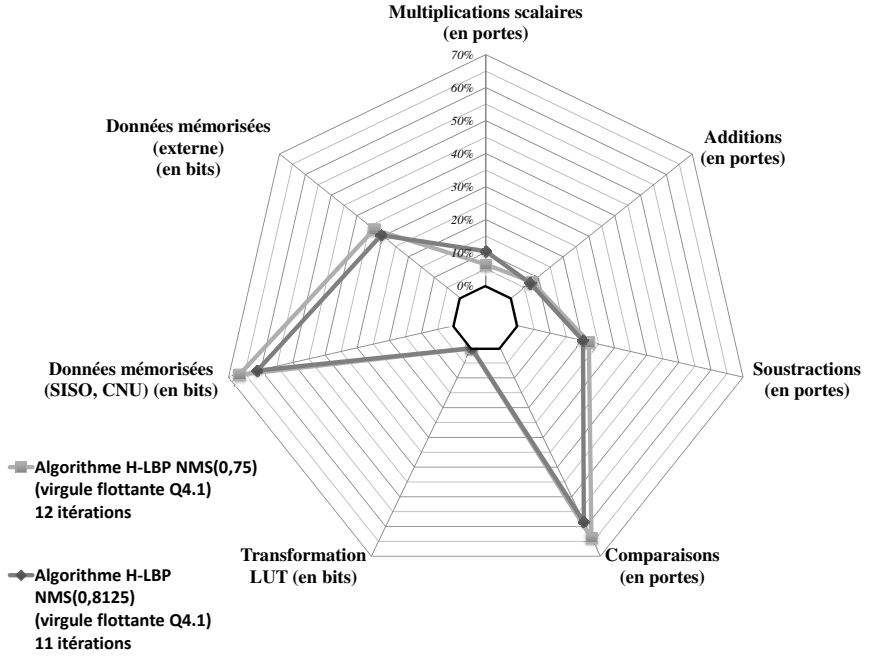


FIGURE 2.11: Complexité de décodage suivant un facteur d'échelonnage pour le décodage de code QC-LDPC

Dans ces conditions, bien que la multiplication scalaire avec le facteur 0.8125 coûte une addition supplémentaire, soit un surcoût total de $1.2.10^6$ portes, le gain d'une itération permet de préserver 10^6 portes logiques sur l'ensemble du décodage. La TABLE 2.8 résume ces gains. L'empreinte de ces algorithmes est visualisée sur la FIGURE 2.12. Sur cette empreinte, le nombre de données mémorisées est uniquement proportionnel au nombre d'itérations. Cette complexité est donc plus favorable pour le facteur 0.8125. La complexité logique est quant à elle complètement compensée par le gain d'une itération. Cette étude montre donc que le choix d'un facteur d'échelonnage plus complexe permet de réduire la complexité globale de décodage à performance constante.

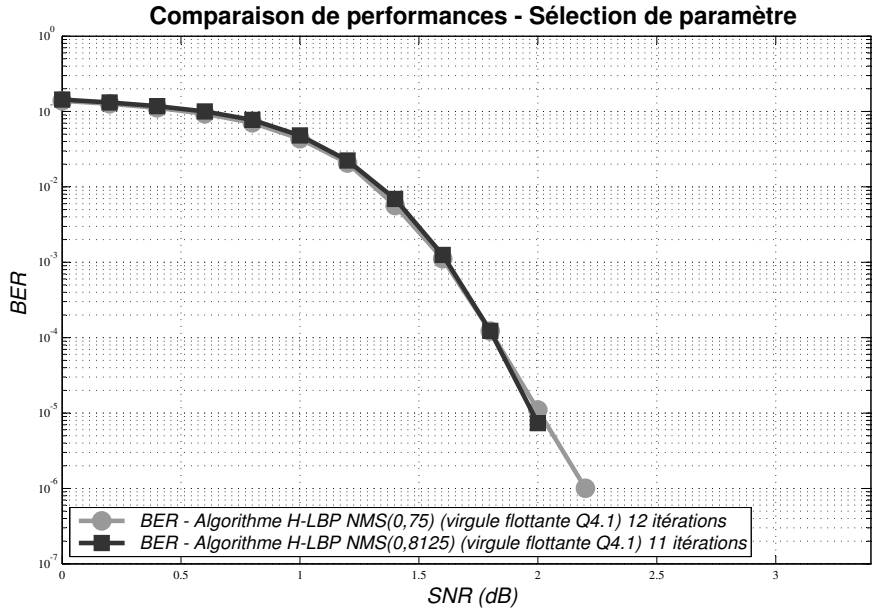


FIGURE 2.12: Performance de décodage suivant un facteur d'échelonnage pour le décodage de code QC-LDPC

2.3 Architectures de décodage de treillis

Le décodeur BCJR est un algorithme de décodage appliqué aux codes en treillis et présentant de bonnes performances BER appliqué aux turbocodes. Cette section établit une architecture de décodage des codes représentés sur un treillis en fonction des caractéristiques de ce treillis. Différentes variantes sont testées pour améliorer les performances de décodage. La complexité globale est également évaluée au moyen des métriques présentées dans la section 2.1.

2.3.1 Architecture des métriques de branche

La première étape consiste à calculer le coût de chaque branche $\gamma(s, s')$. Le coût d'une branche est donné par la probabilité de transition $\Pr\{S_k = s', s_k, p_k | S_{k-1} = s\}$, où s_k représente les informations systématiques et p_k les informations redondantes concernant la transition k. Pour le décodage de codes LDPC, s_k représente la variable c_n impliquée dans une équation spécifique e_m. Les éléments de s_k et p_k sont considérés comme indépendants et identiquement répartis. La probabilité s'exprime suivant la relation (2.22).

$\Pr\{S_k = s', s_k, p_k | S_{k-1} = s\} =$

$$\left(\prod_{s_k^i} \Pr\left\{s_k^i | S_{k-1} = s, S_k = s'\right\}\right) \times \left(\prod_{p_k^i} \Pr\left\{p_k^i | S_{k-1} = s, S_k = s'\right\}\right) \qquad (2.22)$$

Opérations	Facteur 0.75	Facteur 0.8125
Complexité logique		
Additions	2.3M (8%)	2.2M (7%)
Soustractions	6.7M (22%)	6.1M (20%)
Comparaisons	19.3M (64%)	17.7M (6%)
Multiplications scalaires	1.9M (6%)	3.1M (10%)
Total de portes	30.2M (100%)	29.1M (96%)
Stockage mémoire		
Stockage des CNU	3.2M (67%)	2.9M (61%)
Stockage externe	1.6M (33%)	1.4M (31%)
Stockage total	4.7M (100%)	4.3M (92%)

TABLE 2.8: Complexité logique et données mémorisées pour le décodage de la matrice IEEE 802.16m de taille $N = 2304$ et de rendement $R = 1/2$

Le Log-Rapport de Vraisemblance attribué aux opérandes de la relation (2.22) font intervenir les Log-Rapport de Vraisemblance intrinsèques systématiques $\mathrm{L}^c\left(s_k^i\right)$ et redondantes $\mathrm{L}^c\left(p_k^i\right)$ du canal. Dans le cadre de décodage itératif, une information extrinsèque supplémentaire est adjointe aux informations systématiques et noté $\mathrm{L}^e\left(s_k\right)$. Cet élément est détaillé à la fin de cette section.

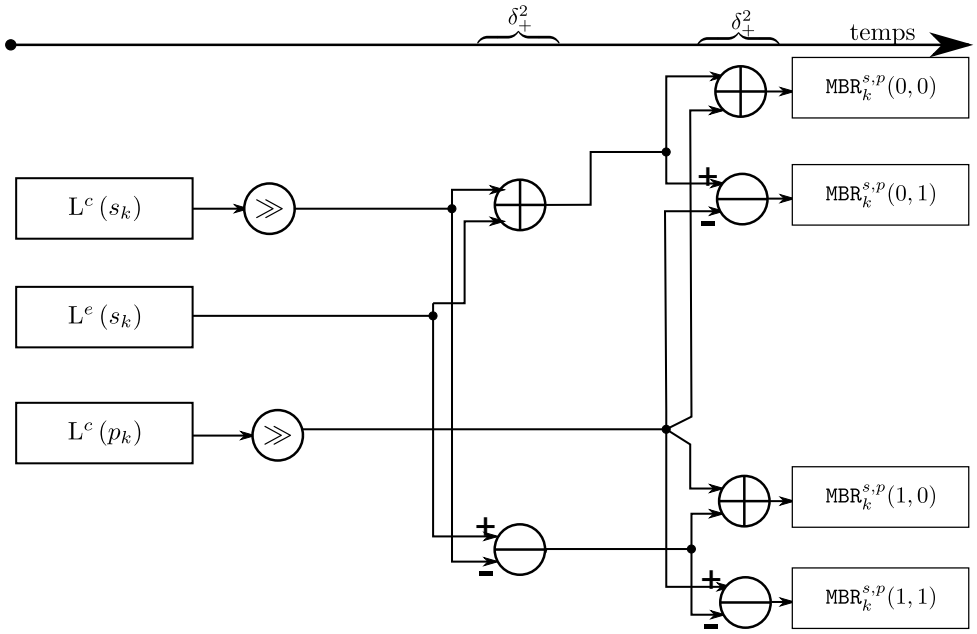
Pour les algorithmes BCJR LogMAP et Max-LogMAP, le calcul des métriques de branche correspond donc à l'addition de deux conditions sur l'information systématique $\mathtt{MBR}_k^s(s_k)$ et l'information de parité $\mathtt{MBR}_k^p(p_k)$ de l'équation (2.23).

$$\mathtt{MBR}_k^{s,p}(s_k, p_k) = \underbrace{\sum_i (-1)^{s_k^i}.\left(\mathrm{L}^c\left(s_k^i\right) + \mathrm{L}^e\left(s_k\right)\right)}_{\mathtt{MBR}_k^s(s_k)} + \underbrace{\sum_i (-1)^{p_k^i}.\left(\mathrm{L}^c\left(p_k^i\right)\right)}_{\mathtt{MBR}_k^p(p_k)} \tag{2.23}$$

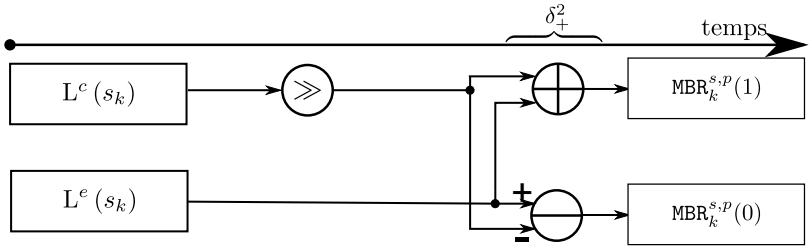
Les conditions systématiques et redondantes de la métrique de branche sont conditionnées par la transition d'état et indépendantes. Chaque association des informations systématiques et redondantes est calculée séparément. La structure du code permet d'associer ensuite les coûts des branches en fonction des états de transition.

Le délai de calcul des métriques de branche est conditionné par la hauteur de l'arbre de calcul associé. Ce délai dépend du nombre d'opérandes et donc de la structure du treillis. La FIGURE 2.13 propose un ordre de calcul pour le cas de métriques de branche d'un code binaire sans parité (FIGURE 2.13a) et dans le cas d'un code binaire avec une parité associée (FIGURE 2.13b).

La TABLE 2.9 détaille la complexité des architectures et leurs délais de propagation en fonction des caractéristiques des codes convolutifs. Cette étape se séquence à travers des pipelines. Dans ce cas, les caractéristiques du code convolutif modifie



(a)



(b)

FIGURE 2.13: Architecture de calcul de métrique de branche dans le cas d'un code binaire sans parité (a) et dans le cas d'un code binaire avec une parité (b)

Opérations	Code convolutif binaire sans parité	Code convolutif binaire avec une parité	Code convolutif double-binaire avec deux parités
Conditions de treillis			
Informations systématiques	1	1	2
Informations redondantes	0	1	2
Complexité			
Divisions par 2	1	2	4
Changements de signe	1	3	0
Additions	1	3	12
Soustractions	1	3	12
Délai de propagation	$\delta_{DYN} + \delta_+^2$	$\delta_{DYN} + 2.\delta_+^2$	$\delta_{DYN} + 3.\delta_+^2$

TABLE 2.9: Complexité et délai de propagation des métriques de branche pour différents codes convolutifs

la latence de calcul mais pas le chemin critique.

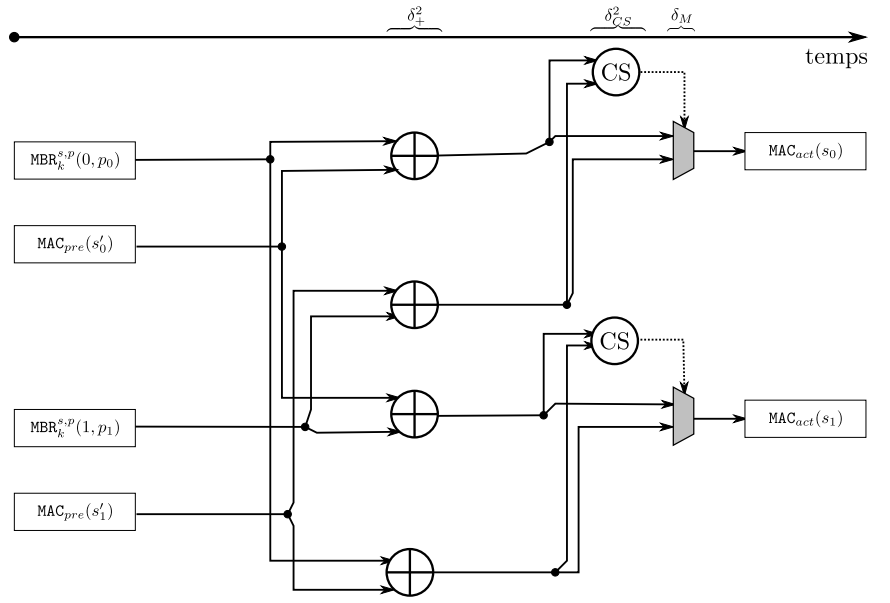
2.3.2 Architecture des métriques cumulées

Le traitement des métriques cumulées nécessite de calculer successivement des informations en suivant le treillis dans les sens *Aller* et *Retour*. Les opérations de calcul des métriques sont équivalentes pour ces deux sens. Les Log-Rapports de Vraisemblance relatifs aux métriques α et β (notées indépendamment $\texttt{MAC}_{act}(s)$ dans la suite) sont conditionnées par les métriques de branche précédemment détaillées ainsi que par les métriques cumulées aux états passés (calcul de α) ou futurs (calcul de β) notées $\texttt{MAC}_{pre}(s')$ dans la suite. Le calcul de ces métriques pour l'algorithme LogMAP se résume par la relation (2.24).

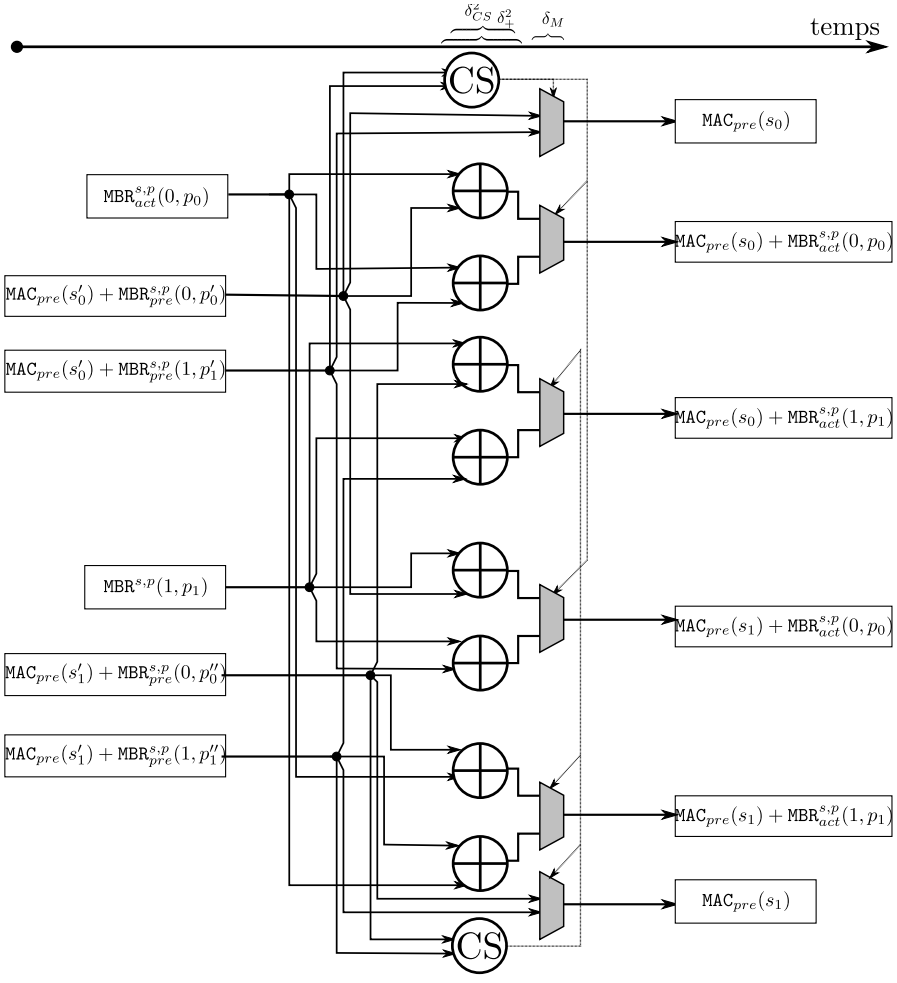
$$\texttt{MAC}_{act}(s) = \max_{s'}{}^{*} \left(\texttt{MBR}_k^{s,p}(s_k, p_k) + \texttt{MAC}_{pre}(s') \right) \tag{2.24}$$

Dans cette section, nous nous intéressons principalement à l'approximation du calcul Max-LogMAP. Dans ce cas, cette opération consiste à effectuer successivement 2 (codes binaires) ou 4 (codes double-binaires) additions suivies d'opérations de comparaison et de sélection. Le résultat obtenu est réinjecté pour le calcul de la section suivante.

Pour un code binaire, ce calcul nécessite 3 opérations séquentielles pour obtenir



(a)



(b)

FIGURE 2.14: Ordonnancement des calculs de métrique cumulée dans l'ordre ACS (a) et CSA (b)

Opérations	Technique ACS	Technique CSA
Complexité pour le traitement de 2 états d'un code binaire		
Additions	2	4
Comparaisons	1	1
Multiplexeurs	1	2
Délai de propagation	$\delta_+^2 + \delta_{CS}^2$	$\max(\delta_+^2, \delta_{CS}^2)$
Complexité pour le traitement de 4 états d'un code double-binaire		
Additions	16	32
Comparaisons	12	12
Multiplexeurs	12	12
Délai de propagation	$\delta_+^2 + 2.\delta_{CS}^2 + 2.\delta_M$	$\max(\delta_+^2, \delta_{CS}^2 + \delta_M) + \delta_{CS}^2 + \delta_M$

TABLE 2.10: Coût du calcul des métriques cumulées en fonction de l'ordre des opérations

le résultat. Cet ordre correspond à une suite d'additions, de comparaisons et de sélections connue comme une méthode ACS (*Addition Comparison Selection*) [84]. Cet ordre de calcul représente un chemin critique pour une architecture de décodage rapide. En effet, le résultat de cet opération est nécessaire au calcul de la métrique cumulée suivante. De ce fait, le délai de propagation ne peut être réduit ni par parallélisme de calcul, ni par séquencement des étapes de calcul.

L'ordre des opérations CSA (*Comparison Selection Addition*) [85] suppose de différer l'étape de comparaison de l'information afin de réduire ce chemin critique. Les étapes de comparaisons et d'additions sont effectuées au même instant, comme indiqué sur la FIGURE 2.14b. Cependant, cet ordre double la complexité de l'étape [86]. La TABLE 2.10 résume l'impact des deux architectures schématisées sur la FIGURE 2.14a et la FIGURE 2.14b.

En ayant préalablement calculé le coût des métriques de branche, la complexité du calcul des métriques cumulées est fonction du nombre d'informations systématiques du code. Le nombre d'états du code n'intervient pas dans le délai de propagation du calcul des métriques cumulées.

L'ordre de calcul ACS est majoritairement utilisé par les architectures de calcul de décodage double-binaire. [87] et [88] favorisent l'ordre de calcul ACS pour des architectures de décodage de codes double-binaire (IEEE 802.16m). [89] utilise un ordre de calcul proche de la métrique CSA pour le décodage de codes double-binaire et adapte cette architecture pour un décodeur LogMAP. Cette architecture est ensuite reprise par [90]. [91] et [70] utilise l'odre ACS pour des architectures de décodage de codes simple binaire. [86] favorise l'ordre CSA pour un code convolutif simple binaire.

2.3.3 Architecture de la décision et de l'information extrinsèque

Les métriques cumulées et les métriques de branche sont utilisées pour calculer l'information de décision finale et l'information extrinsèque associée à l'étape de décodage. Le calcul de la décision *a posteriori* $\mathrm{L}^{a}(s_k)$ suit l'équation (2.25).

$$\mathrm{L}^{a,i}(s_k) = \max_{\substack{s' \longrightarrow s \\ s_k=i}} \big(\alpha_{k-1}(s') + \gamma_k(s,s') + \beta_k(s)\big) - \max_{\substack{s' \longrightarrow s \\ s_k=0}} \big(\alpha_{k-1}(s') + \gamma_k(s,s') + \beta_k(s)\big) \tag{2.25}$$

Cette approche est réalisée en trois phases. Une première étape consiste à calculer toutes les métriques de chemin $\mathtt{DEC}_k^i(s)$ conditionnées à l'information systématique du treillis $s_k = i$ et à un état s (2.26).

$$\mathtt{DEC}_k^i(s) = \alpha_{k-1}(s) + \gamma_k(s,s') + \beta_k(s') \tag{2.26}$$

Lors du parcours du treillis dans le sens *Retour*, la métrique cumulée $\alpha_{k-1}(s)$ est contenue dans une mémoire écrite à la manière d'une LIFO. Avec une architecture de calcul CSA, la métrique $(\gamma_k(s,s') + \beta_k(s'))$ est calculée dans l'étape précédente. Il suffit alors de structurer l'architecture de décodage suivant l'état s. Lors du parcours du treillis dans le sens *Aller*, la somme $(\alpha_{k-1}(s) + \gamma_k(s,s'))$ est déjà accessible pour le calcul des métriques cumulées $\alpha_k(s')$. Il suffit dès lors de structurer l'architecture suivant l'état s'. L'équation (2.26) se généralise suivant (2.27), où $\mathtt{MBR}_k^{s_k,p_k}(i,p_i)$ se réfère à la métrique pré calculée et $\mathtt{MAC}_k^{\mathtt{FIFO}}(s)$ à la métrique cumulée de l'état s à la section k et stockée. La Figure 2.15 affiche alors le chemin le plus court pour cette étape sur un treillis binaire.

$$\mathtt{DEC}_k^i(s) = \mathtt{MAC}_k^{\mathtt{FIFO}}(s) + \big(\mathtt{MAC}_{k-1}(s'_i) + \mathtt{MBR}_k^{s_k,p_k}(i,p_i)\big) \tag{2.27}$$

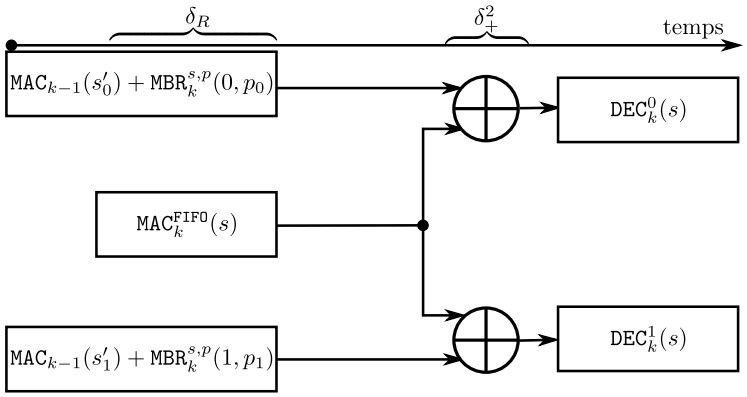


Figure 2.15: Première étape de calcul de la décision et du calcul de l'information extrinsèque

Le type de treillis n'entraîne pas de modification de l'ordonnancement de ces étapes ni de différence de délai de propagation.

Les opérateurs max et max* sont des opérateurs associatifs et commutatifs. De ce fait, l'architecture de décodage définie dans la section 2.1.2 s'applique à cet opérateur. L'architecture choisie consiste à regrouper les états consécutifs. Pour un treillis à 4 états, la valeur maximale est obtenue par l'équation (2.29).

$\forall i \in \mathcal{A}$,

$$\Lambda_k^i = \underset{s' \xrightarrow[d_k=i]{} s}{\overset{*}{\max}} \mathtt{DEC}_k^i(s) \tag{2.28}$$

$\forall i \in \mathcal{A}$,

$$\Lambda_k^i = \underset{i}{\max}{}^* \left(\underset{i}{\max}{}^* \left(\mathtt{DEC}_k^i(0), \mathtt{DEC}_k^i(1) \right), \underset{i}{\max}{}^* \left(\mathtt{DEC}_k^i(2), \mathtt{DEC}_k^i(3) \right) \right) \tag{2.29}$$

La complexité de cette étape de calcul est proportionnelle au nombre d'états S et au cardinal de $\mathcal{A}$. La délai de propagation de cette étape est proportionnelle à $\log_2(S)$. Le cardinal de $\mathcal{A}$ n'entre pas en compte sur ce délai lors de cette phase de calcul. En effet cette opération se parallélise en fonction des éléments systématiques. Pour un treillis à 2 états, seule une étape de comparaison et de sélection est nécessaire, ce qui entraîne une latence de 2 cycles d'horloge. Pour un treillis à 16 états, une architecture parallélisée requiert 15 comparaisons et sélections organisée avec une latence de 8 cycles d'horloge. L'arbre de calcul de l'opération max pour un treillis à 8 états est représenté sur la Figure 2.16. Cette étape permet d'établir la log-probabilité de l'élément i connaissant l'ensemble du treillis.

La dernière phase consiste à retrouver les Log-Rapports de Vraisemblance des éléments systématiques et à en extraire une information extrinsèque.

L'information extrinsèque représente l'ensemble de l'information extraite par le décodage du treillis. En d'autres termes, elle représente la valeur (2.30). Par rapport aux données architecturales, cette équation équivaut à l'opération (2.31)

$$\mathrm{L}^{e,out,i}(s_k) = \log \left(\frac{\Pr\left\{ c_k = i | \boldsymbol{y}_1^{N_m} \right\}}{\Pr\left\{ c_k = i | y_k \right\}} \right) \tag{2.30}$$

$$\mathrm{L}^{e,out,i}(s_k) = \mathrm{L}^{a,i}(s_k) - \mathtt{MBR}_k^{s_k}(i) \tag{2.31}$$

Dans le cas de codes convolutifs binaires, il est plus intéressant de stocker le Log-Rapport de Vraisemblance plutôt que les Log-Vraisemblances. Cette étape nécessite donc une étape de calcul supplémentaire en fonction du cardinal de $\mathcal{A}$. La Figure 2.17 représente cette dernière étape dans le cas d'un treillis binaire (Figure 2.17a) et d'un treillis double-binaire (Figure 2.17b).

Au final, l'étape de décision et ses coûts sont représentés dans la Table 2.11, qui résume la complexité affectée au traitement d'une section de treillis pour un code binaire à 2 et 8 états et pour un code double-binaire à 8 états.

La complexité de l'architecture globale de décodage reprend les données de ce tableau. Il est à noter que la complexité de décodage globale est proportionnelle à la taille du treillis K pour un code à un seul treillis ou de la somme des tailles de treillis T_l sur une itération notée T_{It} pour les turbocodes et autre codes parallèles.

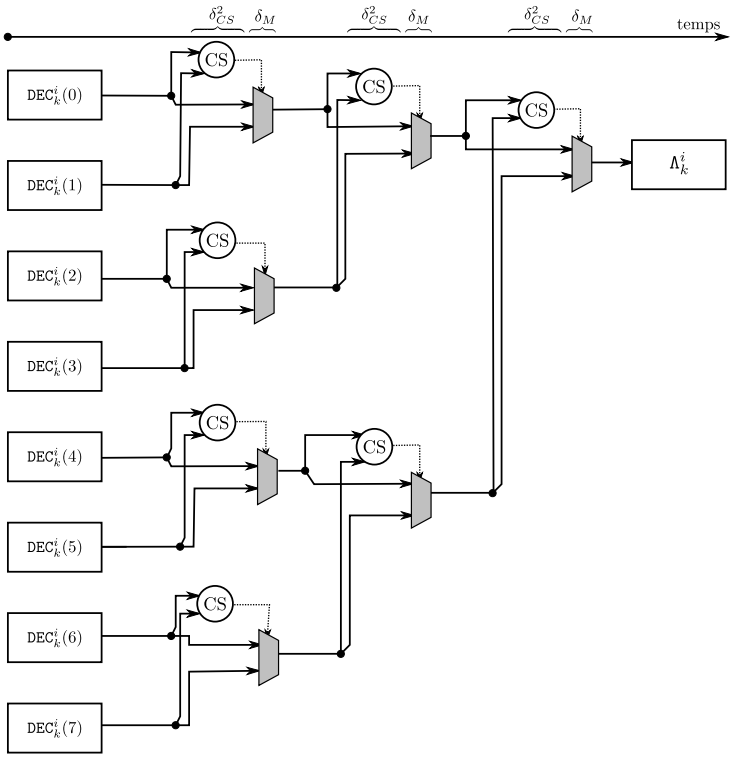


FIGURE 2.16: Seconde étape de calcul de la décision et du calcul de l'information extrinsèque pour code à 8 états

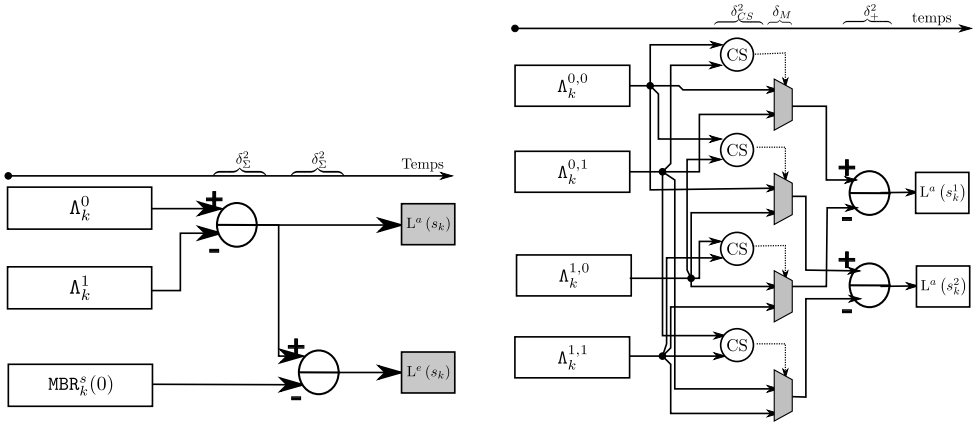
2.3.4 Choix algorithmiques et compromis complexité/performance

2.3.4.1 Échelonnage de l'information et impact sur les performances de décodage

Afin d'améliorer les performances de décodage de l'algorithme Max-LogMAP appliqué sur les turbocodes, un facteur d'échelonnage f_Z est couramment appliqué sur les sorties extrinsèques. L'échelonnage de l'information extrinsèque est alors contraint au nombre d'itérations. Afin d'éviter une complexité trop importante, ces facteurs d'échelonnage sont de l'ordre du multiple de deux, sur la forme (2.32).

$$f_Z = \begin{cases} 0.5 & \text{si } it < 2 \\ 0.75 & \text{si } 2 \leq it < 6 \\ 1 & \text{sinon} \end{cases} \tag{2.32}$$

La complexité due à ces facteurs correspond à des déplacements de la dynamique de l'information extrinsèque (facteurs 0.25, 0.5) où à une addition supplémentaire (facteur 0.75). Ce type d'échelonnage a été testé sur un code binaire à 8 états (Standards 3GPP LTE) de longueur $K = 256$ et de rendement $R = 1/3$. La FIGURE 2.18a fait état de l'amélioration de performance entre le décodage sans échelonnage de l'information et l'algorithme LogMAP, avec 8 itérations. Cette échelonnage a également



(a) (b)

FIGURE 2.17: Dernière étape de calcul de la décision et du calcul de l'information extrinsèque pour code binaire (a) et pour un code double-binaire (b)

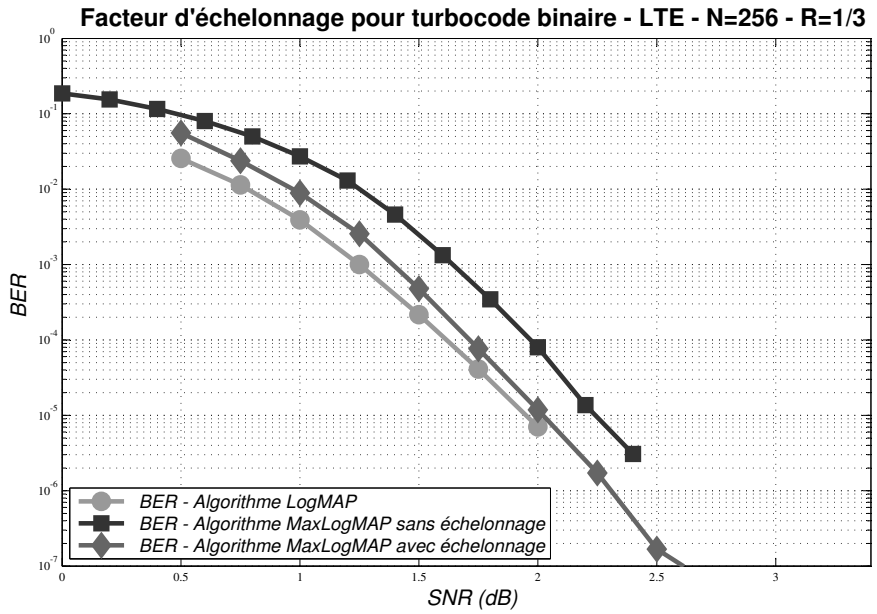
Opérations	**Code convolutif binaire sans parité**	**Code convolutif binaire avec une parité**	**Code convolutif double-binaire avec deux parités**
Additions	7	19	38
Soustractions	7	19	38
Comparaisons	2	14	32
Multiplexeurs	2	14	32

TABLE 2.11: Coût du calcul de la décision suivant les codes convolutifs

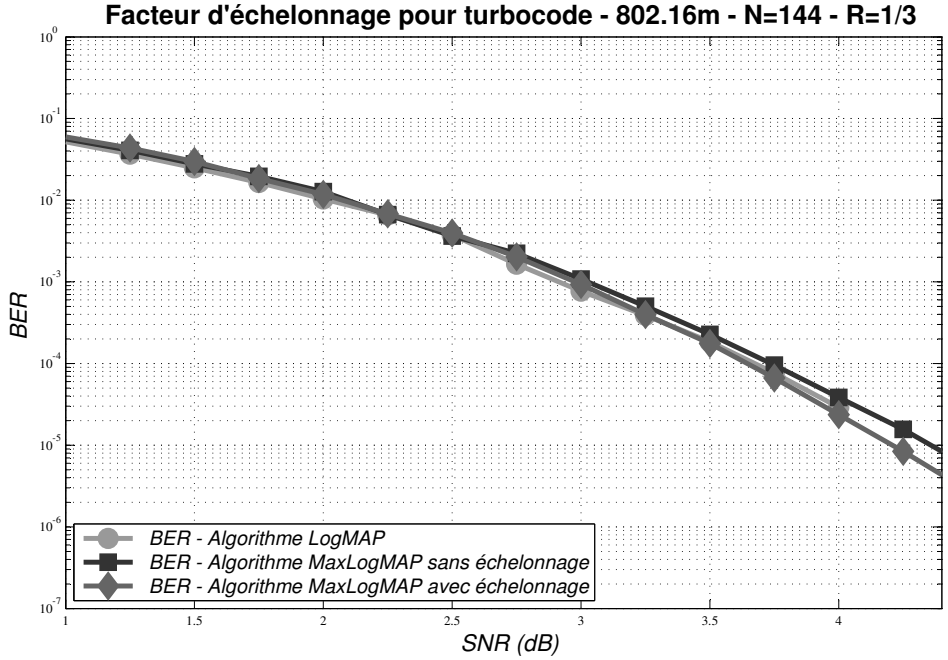
été testé sur un MCS turbocode double-binaire IEEE 802.16m de taille $K = 144$ et de rendement $R = 1/3$. Les mêmes algorithmes et les mêmes facteurs d'échelonnage ont été appliqués. L'amélioration de performance est notable sur ce cas également. Au final, le facteur d'échelonnage permet d'améliorer le décodage de 0.15 dB dans chaque cas de décodage.

2.3.4.2 Décodage suivant le Radix des codes convolutifs

Le chapitre 1 fait état de la transformation en treillis de codes convolutifs binaire suivant la technique Radix-4. Cette technique permet d'appliquer un décodage LogMAP ou Max-LogMAP double-binaire sur un treillis de taille deux fois réduite. Cette technique est intéressante pour le décodage conjoint de codes simple et double-binaire. Dans cette section, on se propose de comparer le rapport entre ces deux approches sur un turbocode binaire du standard 3GPP LTE.



(a)



(b)

FIGURE 2.18: Échelonnage de l'information extrinsèque pour le décodage d'un code simple binaire (a) et double-binaire (b)

La transformation en treillis Radix-4 n'entraîne pas de différences entre le calcul des opérateurs. Ceci est confirmé pour le cas du décodage d'un mot de code de taille $K = 256$ et de rendement $R = 1/3$ sur la FIGURE 2.19. Une architecture correctement conçue n'entraîne pas de différence de performances entre les deux algorithmes de décodage. Le diagramme en toile d'araignée de la FIGURE 2.20 indique dans ce cas un doublement des opérations d'additions et de soustractions qui n'est pas compensé par la réduction de la taille du treillis.

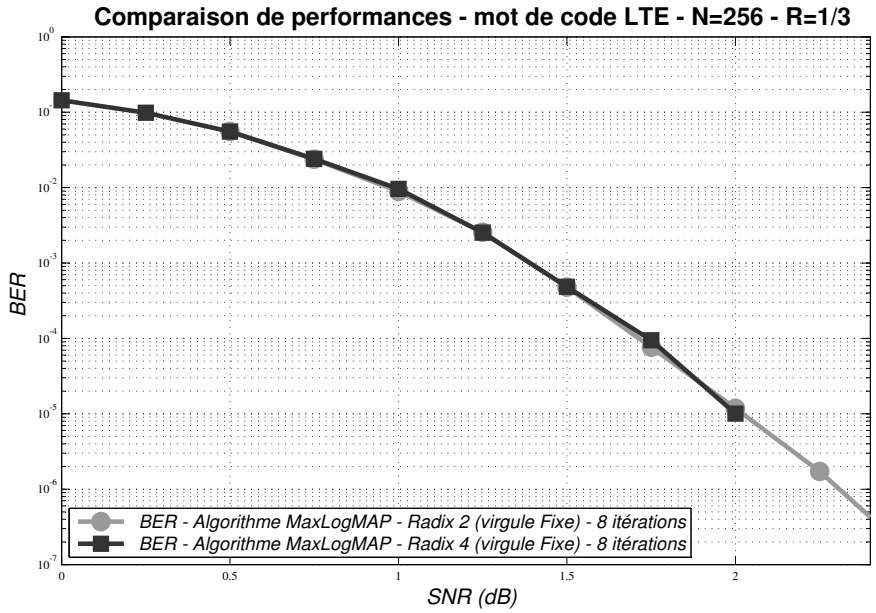


FIGURE 2.19: Impact de la transformation Radix-4 sur les performances de décodage avec l'algorithme Max-LogMAP

La technique de Radix-4 est intéressante pour réduire la latence de décodage du mot de code. Ceci implique cependant de pouvoir transformer le treillis du code convolutifs en un treillis double-binaire.

2.4 Étude comparée des codes correcteurs et de leurs algorithmes de décodage

Une architecture de décodage générique et flexible implantée sur une cible non reconfigurable doit vérifier une forte mutualisation des ressources matérielles. Cette section vise à sélectionner les algorithmes pouvant être intégrés sur une seule et même cible pour le décodage de codes présélectionnés. Pour cela, les codes correcteurs d'erreurs avancés sont comparés à performance équivalente en fonction des algorithmes de décodage présentés dans les sections 2.2 et 2.3.

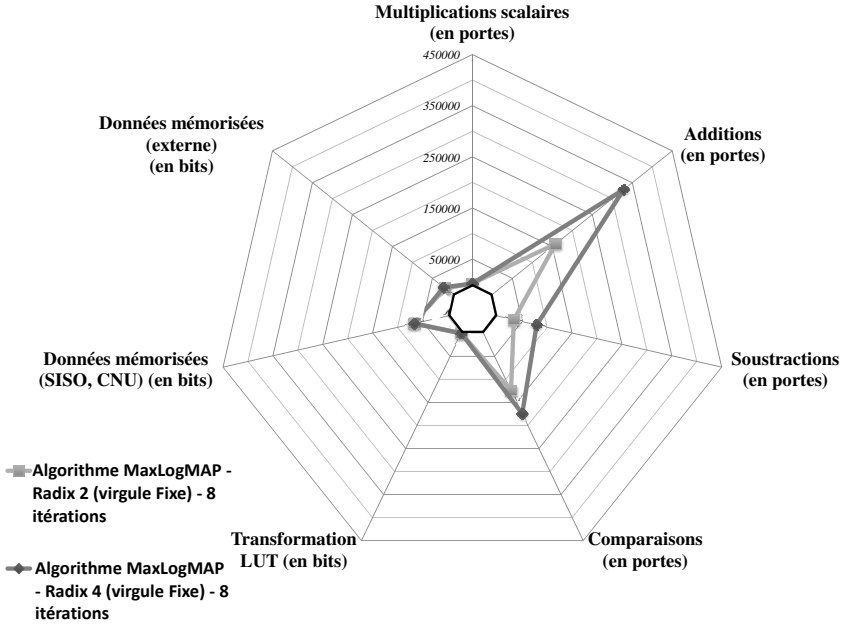


FIGURE 2.20: Impact de la transformation Radix-4 sur la complexité de décodage avec l'algorithme Max-LogMAP

2.4.1 Rapport complexité/performance pour des codes équivalents

Les sections précédentes ont établi des métriques de complexité communes pour caractériser des algorithmes de décodage d'un point de vue matériel, tout en évitant de cibler une technologie particulière. La représentation à partir des diagrammes en toile d'araignée, développée dans la section 2.1.3 permet d'évaluer cette complexité à partir d'une représentation visuelle, en incluant les principaux opérateurs complexes et les requêtes mémoires. Cette complexité peut être normalisée par rapport à un seuil de performance fixé sur différentes stratégies de décodage. Dans cette partie, on cherche à caractériser et à sélectionner des choix algorithmiques pour une structure de décodage conjointe, de sorte que les ressources matérielles soient les plus réutilisées possible. Pour définir cet algorithme d'un point de vue complexité, il est nécessaire de sélectionner des codes équivalents et de représenter leurs coûts de décodage pour une performance fixée. Deux codes sont considérés comme équivalent si ils partagent la même taille de bloc et le même rendement. Le standard IEEE 802.16m favorise des codes LDPC et des turbocodes pour de mêmes profils d'usage. Parmi ceux-ci, il existe une matrice QC-LDPC et un turbocode double-binaire de rendement $R = 1/2$ et de taille de bloc $N = 576$. Dans un premier temps, on repère leurs caractéristiques de décodage à une performance équivalente. D'après les études de complexité réalisées dans la section 2.2.4, l'algorithme NMS de facteur 0.8125 avec un ordonnancement H-LBP est un algorithme performant en termes de décodage et de rapport de complexité. Pour un seuil de fonctionnement BER fixé à

10^{-4} pour 3 dB, cet algorithme requiert 6 itérations de décodage. De ce fait, il est intéressant de vérifier la complexité de calcul associé à cet algorithme sur ce code à 6 itérations.

L'étude de complexité réalisée sur les codes en treillis de la section 2.3.4 montre que l'algorithme Max-LogMAP avec échelonnage de l'information extrinsèque est performant sur ce type de code. Le même seuil de performance est atteint pour 2 itérations de décodage. Ces paramètres sont donc sélectionnés pour une étude comparative entre ces deux choix de décodage.

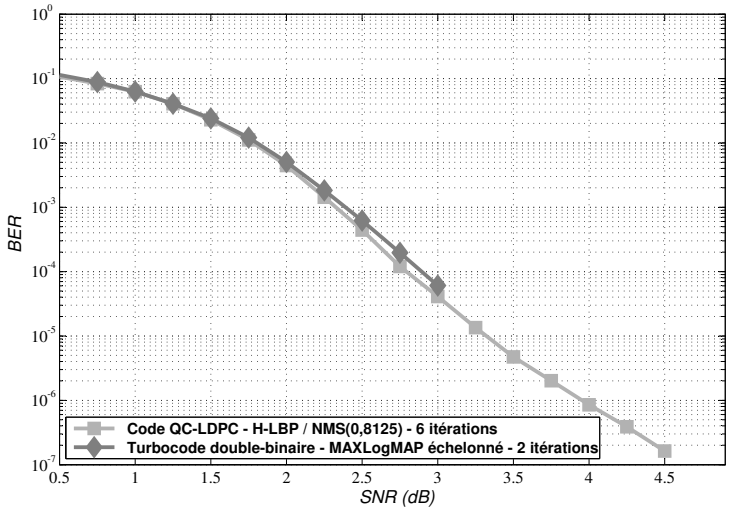
Les performances de décodage associées à ces deux choix de décodage sont représentées sur la FIGURE 2.21a. Ces performances vérifient bien les propriétés recherchées. La FIGURE 2.21b prend en compte la complexité de calcul nécessaire à l'obtention de ces performances. Les empreintes de décodage de ces algorithmes ne sont cependant pas équivalentes. L'algorithme de calcul Max-logMAP nécessite de nombreuses opérations d'additions tandis que l'algorithme H-LBP avec mise à jour NMS favorise une complexité de comparaison de signaux. Les requêtes mémoires sont également plus nombreuses pour le décodage du turbocode principalement dû à un nombre d'itérations plus important pour ce type d'algorithme. La complexité de ces algorithmes montre donc un faible potentiel de mutualisation des ressources matérielles de décodage bien que ces codes soient équivalents. D'autres algorithmes doivent donc être étudiés pour améliorer cette mutualisation.

2.4.2 Algorithme de propagation de croyance appliqué aux turbocodes

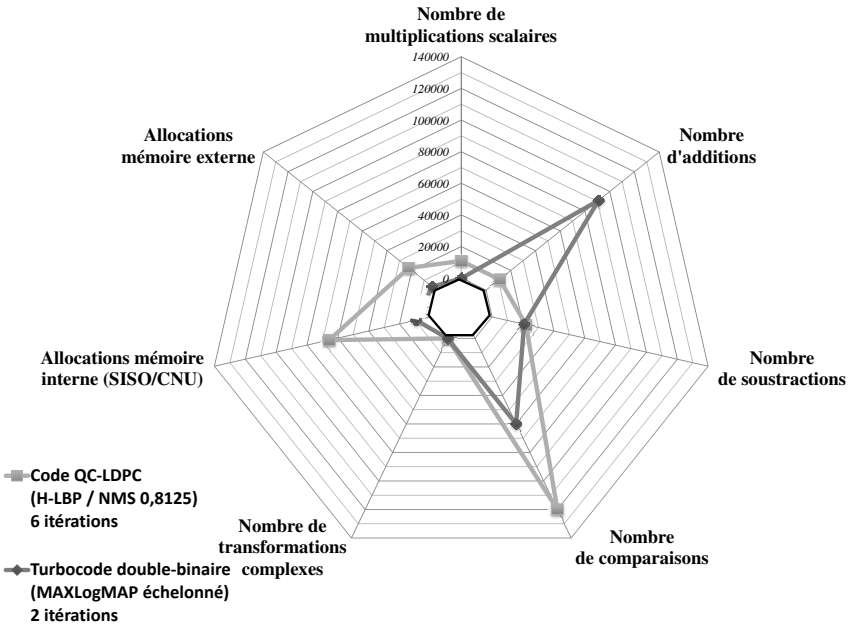
Les algorithmes de calcul suivant la propagation de croyances ont été étudiés pour décoder les codes convolutifs constituant les turbocodes. [92] étudie le turbocode double-binaire du standard IEEE 802.16e. Il reconstitue ainsi une matrice de parité pour un mot de code de K bit avec une matrice de parité de taille $K \times 2.K$. La matrice de parité obtenue est irrégulière de degré de parité variant de 7 à 10 et de degré de variable variant de 3 à 6. [92] développe un algorithme proche de l'algorithme BP pour décoder le code convolutif associé. Cependant, les performances sont dégradées de 2 à 3 dB par rapport à un algorithme Max-LogMAP classique. De plus la matrice obtenue comporte de nombreux cycles courts qui réduisent les performances de décodage. La matrice n'est pas de type QC-LDPC. De ce fait, le parallélisme induit par ce type de code n'est pas applicable pour le décodage de cette structure. Pour ces raisons, cette piste n'est pas étudiée plus en détail dans ce document.

2.4.3 Algorithme BCJR appliqué aux codes LDPC

[93] et [94] représentent les codes LDPC sous la forme de treillis. Dans ce cas, les équations de parité correspondent à une succession de M treillis à deux états de longueur variable d_c. Il n'y a pas de parité associée à une section de treillis. Les algorithmes LogMAP et Max-LogMAP sont donc applicables comme une technique



(a)



(b)

FIGURE 2.21: Comparaison de la complexité (b) de deux codes de taille $N = 576$ et de rendement $R = 1/2$ à performances équivalentes (a)

de mise à jour des nœuds de parité à l'instar des algorithmes SPA et MS.

L'ensemble des contraintes de parité associées à une matrice LDPC est équivalent à une structure en treillis de T_{It} sections vérifiant (2.33).

$$T_{It} = \sum_{m=0}^{M-1} d_c^m = \mathbf{d}^H \tag{2.33}$$

La complexité relative à une itération revient à réutiliser les relations fournies par la TABLE 2.11.

Cependant, l'usage classique de l'échelonnage n'est pas efficace avec un algorithme BCJR sur le décodage des équations de parité d'un code LDPC. En effet, l'information de décision nécessite d'être réutilisée par le décodeur suivant, tandis que l'information extrinsèque en sortie du cœur de décodage SISO donne une information extrinsèque sur l'itération, qui doit être retranchée à l'itération suivante. De ce fait, échelonner cette information revient à réduire le facteur limitant l'auto-confirmation, ce qui entraîne des auto-confirmations évoluant d'itération en itération. Cet échelonnage a été testé sur une matrice du standard IEEE 802.11n de taille $N = 648$ et de rendement $R = 1/2$. La FIGURE 2.22 montre l'impact itération par itération d'un échelonnage de l'information extrinsèque basé sur l'échelonnage turbocode (2.32). L'échelonnage a été dimensionné suivant la rapidité de convergence du code LDPC (pour un décodage H-LBP *Min-Sum*) en fonction de l'échelonnage (2.34).

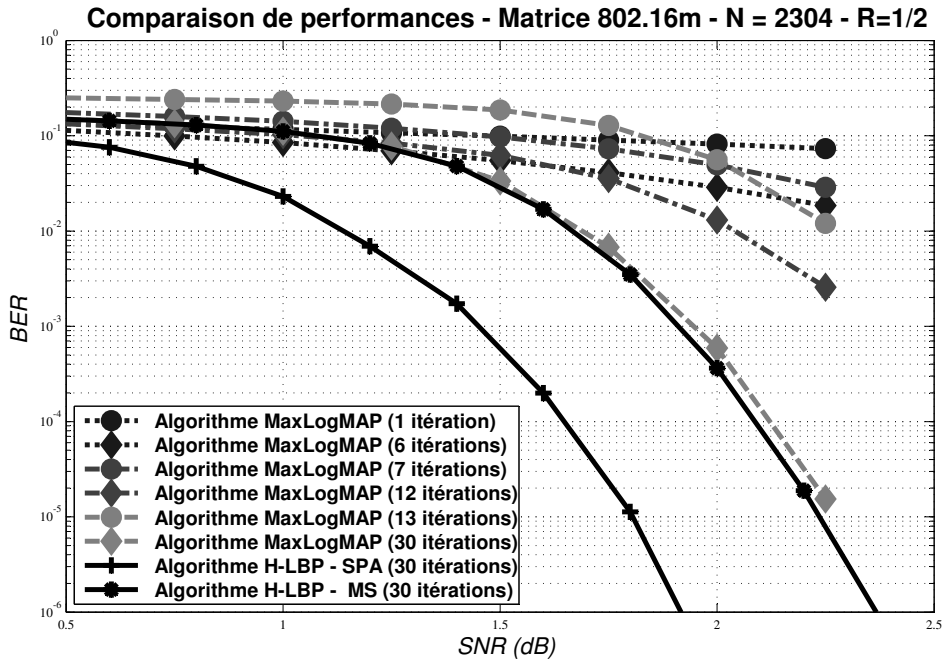


FIGURE 2.22: Application de l'échelonnage de l'information extrinsèque avec les facteurs courants du décodage turbocode appliqué au décodage d'un code QC-LDPC

$$f_Z = \begin{cases} 0.25 & \text{si } it < 2 \\ 0.5 & \text{si } 2 \leq it < 6 \\ 0.75 & \text{si } 6 \leq it < 12 \\ 1 & \text{sinon} \end{cases} \tag{2.34}$$

En définitive, la méthode d'échelonnage courante du décodage turbocode ne doit pas être effectuée uniquement sur l'information extrinsèque mais également sur l'information de décision, qui est reprise lors du décodage.

Cette normalisation s'effectue sur le dernier bloc de décision. Il suffit alors d'incorporer un facteur de normalisation f_Λ sur la décision finale et de poser $f_\Lambda = f_Z^{-1}$. Dans ce cas, l'échelonnage permet d'approcher les performances de décodage atteintes par les algorithmes H-LBP NMS avec les facteurs f_Λ. La relation (2.35) donne les couples utilisés dans l'expérimentation de cette architecture de décodage.

$$\begin{array}{rcl} f_\Lambda = 0.75 & \Leftrightarrow & f_Z = 4/3 \\ f_\Lambda = 0.80 & \Leftrightarrow & f_Z = 1.25 \end{array} \tag{2.35}$$

La Figure 2.23 montre l'impact de l'échelonnage de l'information de décision et extrinsèque sur une matrice LDPC du standard IEEE 802.11n de taille $N = 648$ et de rendement $R = 1/2$. Sur ces courbes, on remarque également que les facteurs d'échelonnage attribués au décodage turbocode ne sont pas aussi performants que de sélectionner des échelonnages indépendants du nombre d'itérations.

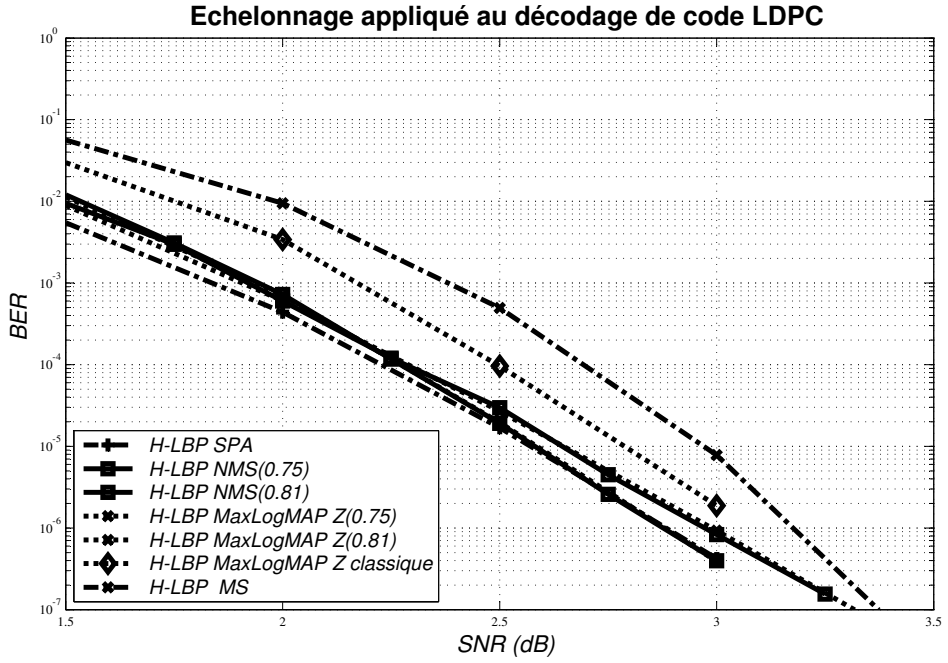


Figure 2.23: Étude de l'échelonnage statique de l'information extrinsèque pour le décodage du code LDPC IEEE 802.11n de taille $N = 648$ et de rendement $R = 1/2$

2.4.4 Rapport complexité/performance avec algorithmes équivalents

Le choix entre différentes stratégies de codage (turbocode ou LDPC) peut être envisagé d'un point de vue de la performance de ces codes suivant le type de propagation ou en fonction de leur rapport de complexité. Dans cette section, on propose d'observer deux codes de caractéristiques de codage équivalent, soit des rendements et des tailles de codes équivalents. Cette caractéristique est obtenue par exemple par deux MCS du standard IEEE 802.16m qui définit aussi bien un codage turbocode ou un codage QC-LDPC.

Pour établir un rapport de performance, il suffit de fixer les caractéristiques de codage de rendement $R = 1/2$ et de taille de bloc $N = 1920$. Pour ce type de code, il existe deux MCS du standard IEEE 802.16m dont l'un privilégie un turbocode et l'autre un code QC-LDPC. Bien que le terme d'itérations n'aie aucune signification pour comparer ces deux stratégies de codage, on propose de fixer le nombre d'itérations nécessaires pour obtenir des performances équivalentes et d'en extraire la complexité de décodage pour ces deux types de codes. Les algorithmes de calcul utilisés sont un algorithme de calcul H-LBP avec mises à jour Max-LogMAP pour les codes QC-LDPC et un algorithme Max-LogMAP pour le turbocode. Pour ce cas d'usage, le BER est fixé à 3.10^{-5} pour un SNR de 2 dB. Le nombre d'itérations turbocode est fixé à 4 tandis que 11 itérations des codes QC-LDPC sont nécessaires. La Figure 2.24 montre la performance de décodage pour ces deux codes. Le diagramme en toile d'araignée associé représenté sur la Figure 2.25 montre l'impact de la complexité de calcul pour une quantification sur 6 bits (Q5.1). Le diagramme représente la complexité de calcul normalisée par nombre de porte.

2.5 Conclusion

Les algorithmes de calcul des mises à jour de codes correcteurs d'erreurs se caractérisent par l'utilisation de plusieurs opérateurs logiques. Ce chapitre présente une étude de la complexité et du temps de calcul de plusieurs algorithmes de décodage en fonction de critères architecturaux. Tout en évitant de sélectionner une technologie de décodage particulière, ce chapitre propose une représentation graphique de la complexité architecturale et permet d'élaborer des critères de décision sur un algorithme ou un paramètre en fonction de sa complexité littérale ou en fonction de sa complexité matérielle. Cet outil a ensuite été réutilisé pour sélectionner un algorithme de décodage permettant de décoder deux codes distincts avec deux algorithmes différents de façon à obtenir des performances équivalentes.

Cette étude met en évidence le choix de décodage de codes QC-LDPC suivant l'algorithme de propagation de croyance avec ordonnancement horizontal et décodage en treillis avec un facteur d'échelonnage. Ce choix algorithmique permet à deux codes QC-LDPC et turbocode d'utiliser le même type d'opérateurs et de ressources. Le choix de cet algorithme permet alors de mutualiser efficacement les ressources matérielles pour le décodage de codes QC-LDPC et de turbocodes. Cependant, pour

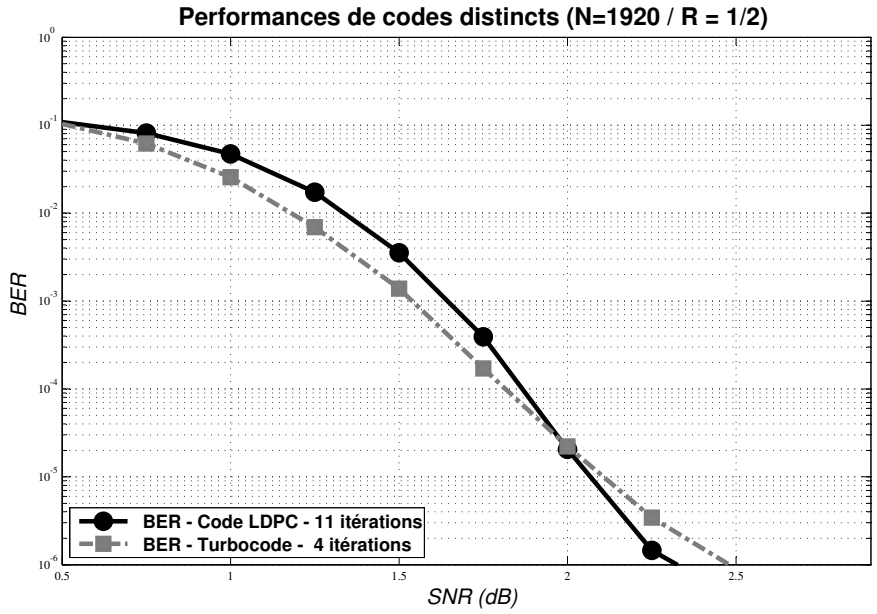


FIGURE 2.24: Performance de décodage pour des codes équivalents avec un algorithme de mise à jour équivalent (BCJR)

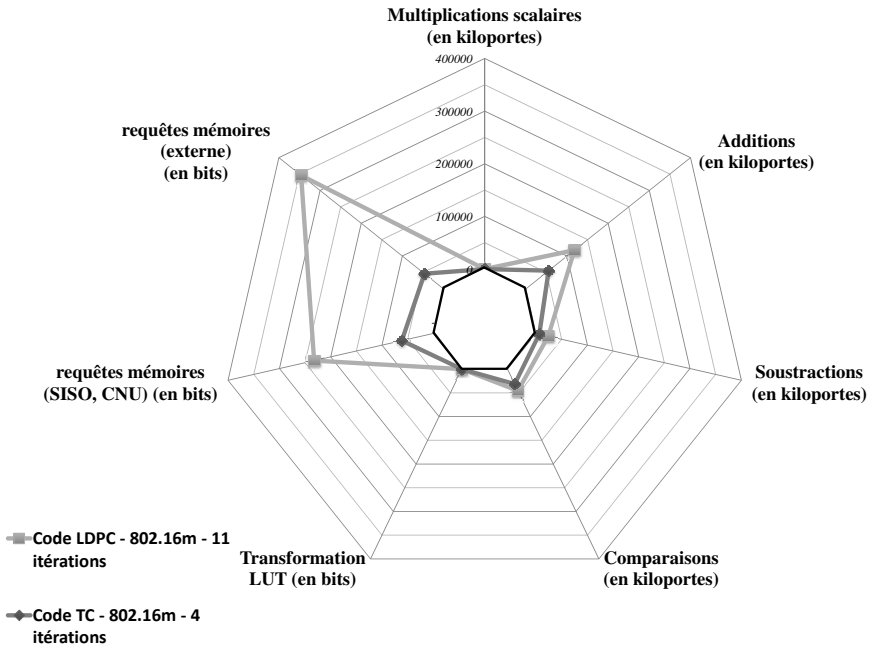


FIGURE 2.25: Complexité de décodage pour des codes équivalents avec un algorithme de mise à jour équivalent (BCJR)

arriver à une mutualisation efficace, les codes QC-LDPC doivent être étudiés afin d'établir une structure en treillis avec un entrelaceur simple, et la structure de décodage des turbocodes doit également être adaptée aux contraintes d'un usage de décodage QC-LDPC.

Chapitre 3

Généralisation de treillis pour un décodage conjoint

Le chapitre 2 analyse la pertinence de la sélection d'un algorithme BCJR pour décoder conjointement des codes QC-LDPC et des turbocodes. Dans ce chapitre, on s'intéresse à la forme de ces différents codes afin d'établir une représentation en treillis commune. Cette représentation offre le double avantage de permettre la mutualisation de certains blocs de calculs ainsi que d'offrir une stratégie commune d'ordonnancement sur une itération de décodage.

Dans un premier temps, les treillis associés aux turbocodes sont étudiés pour en extraire une représentation commune. Une structure de treillis générique y est étudiée et les codes constituant les turbocodes y sont classifiés en fonction de ce type de représentation. Dans un second temps, une étude des codes QC-LDPC est effectuée afin de faire correspondre à une matrice QC-LDPC un treillis équivalent en définissant dans ce cas les fonctions d'entrelacement. Dans un troisième temps, les treillis associés aux deux types de codes seront détaillés en fonction des caractéristiques de calcul d'un algorithme BCJR. Ainsi, cette partie utilise les caractéristiques d'ordonnancement de treillis et étudie les propriétés des entrelaceurs pour extraire un parallélisme de calcul pour chaque code. Enfin, l'ordre de calcul des opérations associées au décodage QC-LDPC est étudié en prenant en compte les contraintes matérielles de décodage pour améliorer les débits de décodage sur ce type de structure.

3.1 Désentrelacement des transitions de sections de treillis des codes convolutifs

Il faut retenir du chapitre 2 que l'adaptation de l'algorithme BCJR au décodage de turbocodes et de codes QC-LDPC présente un bon rapport entre les performances de décodage et sa complexité. Cependant, afin de favoriser la mutualisation de décodage, il convient de revoir l'architecture même des treillis turbocodes et LDPC pour obtenir un degré de mutualisation important. L'étude des treillis permet de favoriser la mutualisation des ressources de calcul pour une architecture commune. Les treillis associés aux codes constituants RSC binaires et double-binaires des turbocodes sont abordés dans cette section pour répondre à cette problématique.

3.1.1 Représentation d'une section de treillis générique à deux états

L'algorithme BCJR classique s'articule à partir d'une représentation en treillis. Ce treillis trace les contraintes des codes et souligne l'architecture de certaines opérations.

Cette partie propose de reprendre les éléments des codes convolutifs afin de les décortiquer en éléments les plus indépendants possibles. Pour cela, on propose de définir une structure de treillis à deux états dont toutes les transitions associées à un élément systématique $s_k = i$ engendrent le même élément de parité p_k^i. De ce fait l'information de parité est identique sur chaque chemin engendré par l'information systématique $s_k = i$. Cette propriété est vérifiée par exemple pour des treillis à deux états sans information de parité. Nous proposons dans un premier temps d'étendre cette représentation sans fixer ni les états initiaux, ni les états finaux. De ce fait, on définit un couple d'états initiaux $\{S_{in}^0, S_{in}^1\}$ et on suppose qu'il existe uniquement deux configurations systématiques/parités $\{s_k = 0, p_k^0\}$ et $\{s_k = 1, p_k^1\}$ entraînant des transitions de treillis menant à deux états finaux $\{S_{out}^0, S_{out}^1\}$.

Au niveau du décodage, la mise à jour des métriques cumulées ne fait appel qu'à deux métriques de branches notées alors $\mathtt{MBR}_k^{s,p}(i, p_i)$.

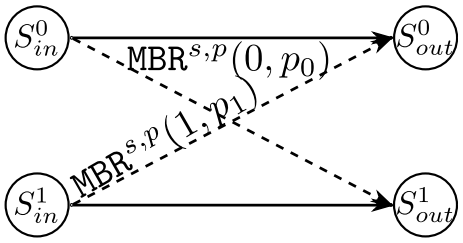


FIGURE 3.1: Représentation d'une section de treillis générique à deux états

La FIGURE 3.1 représente cette *section de treillis générique à deux états*. Cette structure permet de généraliser une architecture de calcul des métriques cumulées en regroupant les opérations par sous-groupes de deux états pour le décodage de codes spécifiques. Il reste encore à déterminer les codes constituant les turbocodes

pour lesquels cette représentation s'applique et à étudier les codes QC-LDPC pour établir une représentation commune.

3.1.2 Découpe des codes RSC binaires

Les turbocodes standardisés font appel à des codes convolutifs récursifs. Dans cette partie, la forme de ces codes est étudiée pour une représentation sur une structure de treillis générique. La proposition 3.1.1 est une condition suffisante permettant aux codes constituant les turbocodes d'être décrits sous forme de sections de treillis générique à deux états.

Proposition 3.1.1. *Un code RSC binaire à $S = 2^\nu$ états définit $\frac{S}{2}$ groupes de transitions indépendantes partageant uniquement deux configurations de métriques de branche si :*

1. *L'information systématique courante n'influence que la valeur de la première mémoire à l'instant k.*
2. *L'information de parité résultante prend en compte la valeur de la dernière mémoire.*

Démonstration. Les codes RSC sont définis par un ensemble de n fonctions de transfert $\boldsymbol{H}(z)$ (3.1) dont chaque polynôme numérateur $\boldsymbol{G^i}(z)$ et le polynôme dénominateur $\boldsymbol{N}(z)$ sont de degré maximal ν.

$$\boldsymbol{H}(z) = \left[1,\ \frac{\boldsymbol{G^1}}{\boldsymbol{N}(z)},\ ...\frac{\boldsymbol{G^n}(z)}{\boldsymbol{N}(z)},\right] \tag{3.1}$$

La forme du polynôme est représentée par (3.2).

$$\boldsymbol{G^i}(z) = \sum_{j=1}^{\nu} g_j^{\mathbf{i}}.z^{-j} \tag{3.2}$$

On suppose que le treillis est initialisé à l'état S_{in}^0. Les blocs mémoires du code convolutif associé prennent alors les valeurs binaires $\{M_1^{k-1}, M_2^{k-1}, ..., M_\nu^{k-1}\}_2$. Le code envisagé étant un code récursif, la valeur de la rétroaction est notée R_k. La formule (3.3) permet de définir les informations de parité.

$$p_k^{\mathbf{i}} \quad = \quad s_k \oplus R_k \bigoplus_{j=1}^{\nu} g_j^{\mathbf{i}}.M_j^{k-1} \tag{3.3}$$

D'après la condition 1 de la proposition 3.1.1, l'information systématique n'influence que la première mémoire M_1^k. De ce fait, le système (3.4) caractérise l'état de sortie $S_0^{out} = \{M_1^k, M_2^k, ..., M_\nu^k\}_2$.

$$\begin{cases} M_1^k &= s_k \oplus R_k \\ M_2^k &= M_1^{k-1} \\ & ... \\ M_\nu^k &= M_{\nu-1}^{k-1} \end{cases} \tag{3.4}$$

L'information systématique $\overline{s_k}$ permet d'établir les redondances avec la relation (3.5).

$$\begin{array}{rcl} {p'}_k^{\mathbf{i}} & = & \overline{s_k} \oplus R_k \bigoplus_{j=1}^{\nu} g_j^{\mathbf{i}}.M_j^{k-1} \\ {p'}_k^{\mathbf{i}} & = & \overline{p_k^{\mathbf{i}}} \end{array} \tag{3.5}$$

Le système d'équations (3.6) caractérise alors l'état de sortie de la transition $S_1^{out} = \{{M'}_1^k, {M'}_2^k, ..., {M'}_\nu^k\}$.

$$\left\{ \begin{array}{rcccl} {M'}_1^k & = & \overline{s_k} \oplus R_k & = & \overline{M_1^k} \\ {M'}_2^k & = & M_1^{k-1} & = & M_2^k \\ & & ... & & \\ {M'}_\nu^k & = & M_{\nu-1}^{k-1} & = & M_\nu^k \end{array} \right. \tag{3.6}$$

Supposons qu'il existe un état initial $S_1^{in} = \{{M''}_1^{k-1}, {M''}_2^{k-1}, ..., {M''}_\nu^{k-1}\}$ dont l'information systématique $\overline{s_k}$ conduise le système à l'état $S_0^{out} = \{M_1^k, M_2^k, ..., M_\nu^k\}$. Nécessairement, l'état d'arrivée S_0^{out} vérifie (3.7).

$$\left\{ \begin{array}{rcl} M_1^k & = & \overline{s_k} \oplus {R''}_k \\ M_2^k & = & {M''}_1^{k-1} \\ & ... & \\ M_\nu^k & = & {M''}_{\nu-1}^{k-1} \end{array} \right. \tag{3.7}$$

La parité associée à cette transition correspond alors à la relation (3.8).

$${p''}_k^{\mathbf{i}} = \overline{s_k} \oplus {R''}_k \bigoplus_{j=1}^{\nu} g_j^{\mathbf{i}}.{M''}_j^{k-1} \tag{3.8}$$

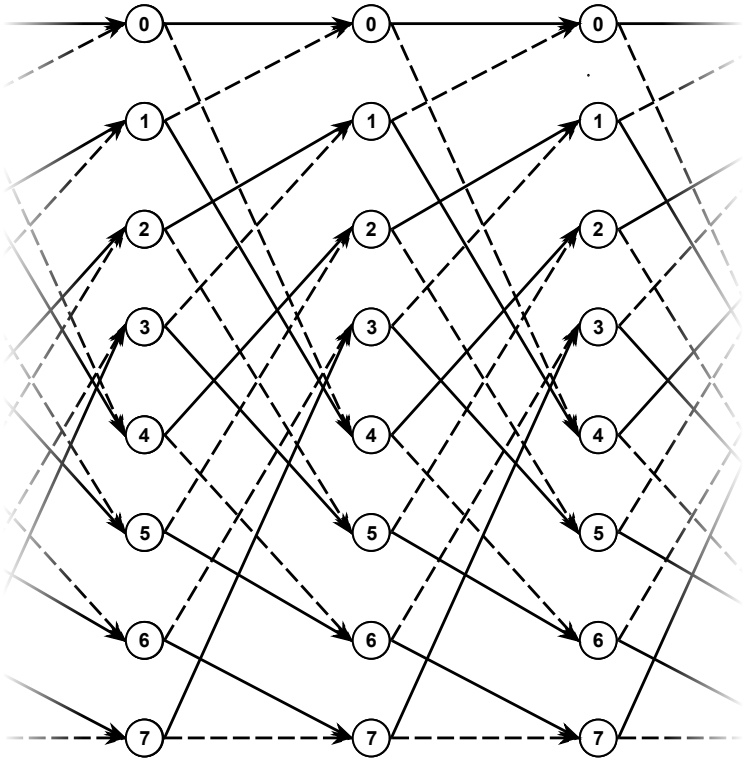
Comme les systèmes (3.4) et (3.7) sont équivalents, on en déduit (3.9).

$$\left\{ \begin{array}{rcl} s_k \oplus R_k & = & \overline{s_k} \oplus {R''}_k \\ {M''}_1^{k-1} & = & M_1^{k-1} \\ & ... & \\ {M''}_{\nu-1}^{k-1} & = & M_{\nu-1}^{k-1} \end{array} \right. \tag{3.9}$$

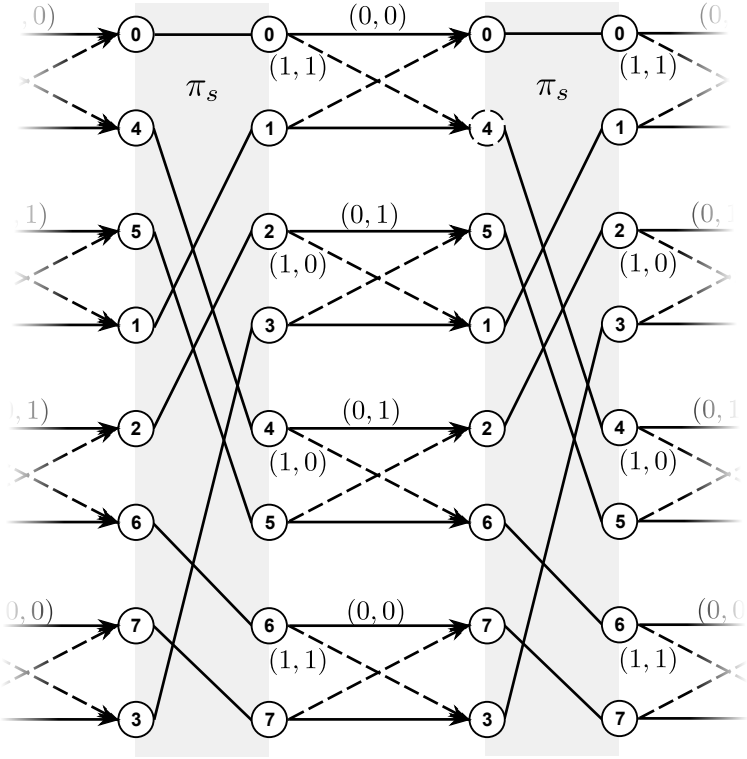
On déduit de la première équation du système (3.9) que ${R''}_k = \overline{R_k}$. On déduit alors de (3.8) que la parité associée à la transition vérifie (3.10).

$${p''}_k^{\mathbf{i}} = \overline{p_k^{\mathbf{i}}} \Leftrightarrow g_\nu^{\mathbf{i}} = 1 \tag{3.10}$$

Une démonstration similaire montre que S_{in}^0 conduit à l'état S_{out}^1 avec $\overline{s_k}$ et produit la parité p^1. Les codes RCS binaires qui vérifient les conditions 1 et 2 de la proposition 3.1.1 permettent bien une découpe en sections de treillis à deux états généralisés représentée sur la FIGURE 3.1. □



(a)



(b)

FIGURE 3.2: Diagramme en treillis du code convolutif 3GPP LTE en représentation classique (a) et suivant une structure de treillis générique (b)

La propriété 3.1.1 est vérifiée pour la plupart des codes convolutifs binaires constituant les turbocodes binaires. Cette représentation est applicable aux standards UMTS [21], LTE [22], cdma2000 [23] ou CCSDS [20]. Le treillis du code constituant le standard LTE est représenté sur la FIGURE 3.2a dans son arrangement classique. La représentation du même code en désentrelaçant le treillis suivant le treillis générique à deux états introduit à la section 3.1.1 amène à la *structure de treillis générique* représentée sur la FIGURE 3.2b. Une section de treillis est alors constituée d'un schéma de transitions uniformes suivi par une fonction d'entrelacement π_s qui permet alors de réordonner les états pour la prochaine section de treillis.

3.1.3 Généralisation aux cas double-binaires

La description du treillis sous la forme de sections de treillis génériques à deux états permet de définir des éléments de base d'un treillis, à laquelle correspond un bloc de calcul ACS ou CSA défini dans le chapitre 2. Cette description est adaptée aux codes binaires. Les codes convolutifs non-binaires ne garantissent pas ce genre de découpe *a priori*. Cependant, les codes double-binaires offrent des degrés de parallélisme de treillis suivant une structure analogue.

Dans le cas d'un turbocode double-binaire, il faut considérer la valeur d'un couple, et non d'une variable binaire. Si une valeur du couple (s_k^1, s_k^2) est temporairement fixée, comme par exemple l'élément s_k^2, le treillis double-binaire se scinde en deux treillis binaires superposés dont l'un représente l'ensemble des transitions (s_k^1, s_k^2) et $(\overline{s_k^1}, s_k^2)$ et l'autre l'ensemble des transitions $(s_k^1, \overline{s_k^2})$ et $(\overline{s_k^1}, \overline{s_k^2})$. Si ces deux treillis binaires vérifient une structure de treillis générique à deux états, une mutualisation des blocs de calculs ACS ou CSA est envisageable.

Proposition 3.1.2. *Les codes constituant les turbocodes associés aux standards IEEE 802.16m [17], DVB-RCS [24], DVB-RCT [25] et DVB-RCS2 [27] permettent une description en treillis génériques à deux états.*

Démonstration. Pour ces codes, si s_k^2 est fixé, alors la variable s_k^1 n'influence que la première mémoire M_1^k. En effet, le système (3.11) décrit le passage d'un état au suivant, où $\boldsymbol{B}(z) = \oplus_{j=0}^{\nu} g_j^{\mathbf{b}}.z^j$ le polynôme d'influence de la variable s_k^2.

$$\left\{ \begin{array}{rcl} M_1^k & = & s_k^1 \oplus s_k^2 \oplus R_k \\ \forall m' \in [\![2;\nu]\!], \quad M_{m'}^k & = & M_{m'-1}^{k-1} \oplus g_\nu^{\mathbf{b}}.s_k^2 \end{array} \right. \tag{3.11}$$

De ce fait, une entrée de la machine avec la valeur $\overline{s_k^1}$ renvoie les relations (3.12).

$$\begin{array}{rcccl} R'_k & = & \bigoplus_{j=1}^{\nu} g_j^{\mathbf{r}}.M_j^{k-1} & = & R_k \\ p_k^i & = & \overline{s_k^1} \oplus s_k^2 \oplus R_k \bigoplus_{j=1}^{\nu} g_j^{\mathbf{i}}.M_j^{k-1} & = & \overline{p_k^i} \\ M'^k_0 & = & \overline{s_k^1} \oplus s_k^2 \oplus R_k & = & \overline{M_0^k} \\ \forall m' \in [\![2;\nu[\![\quad M'^k_{m'} & = & M_{m'-1}^{k-1} \oplus g_\nu^{\mathbf{b}}.b_k & = & M_{m'}^k \end{array} \tag{3.12}$$

En fixant la valeur de la seconde information systématique s_k^2, on se retrouve avec les conditions de deux treillis binaires superposés, dont chacun vérifie les propriétés de la proposition 3.1.1.

□

Ce treillis se représente alors sur deux niveaux. Cette superposition implique qu'une étape de calcul supplémentaire est nécessaire à chaque jonction de treillis, afin de regrouper les états dédoublés. En effet, le calcul des métriques cumulées fait intervenir non pas 2 mais 4 métriques de branche. Cette description permet de hiérarchiser le calcul des métriques de branche pour optimiser la mutualisation des ressources et de réduire les chemins entre les opérateurs.

Les codes constituant les turbocodes des standards IEEE 802.16m, DVB-RCS, DVB-RCT, et DVB-RCS2 vérifient cette propriété. De ce fait, ils sont descriptibles à travers une structure de treillis générique. Le code constituant le turbocode HomePlug AV [28] ne répond pas aux conditions de la proposition 3.1.1. Le treillis classique de ce code est représenté sur la FIGURE 3.3a. Le treillis générique à deux états s'exploite cependant sur ce code en fixant sur un premier niveau les couples de valeurs (s_k^1, s_k^2) et $(\overline{s_k^1}, s_k^2)$. Ce niveau présente alors une structure de treillis uniforme tout comme pour les codes convolutifs binaires. Pour établir l'ensemble des transitions, les deux autres valeurs de couple décrivent un second niveau de transitions uniformes. La structure équivalente est reprise sur la FIGURE 3.3b. Cependant, les deux niveaux de transitions imposent des doublons d'états qu'il convient de regrouper pour prendre en compte la caractéristique double-binaire du code. Cette étape est représentée sur le schéma par un bloc grisé. La fonction d'entrelacement π_s permet de réorganiser les états du treillis pour la section suivante.

Dans cette partie, nous avons proposé une *structure de treillis générique* qui s'adapte à l'ensemble des turbocodes répertoriés dans la TABLE 1.2 du chapitre 1.

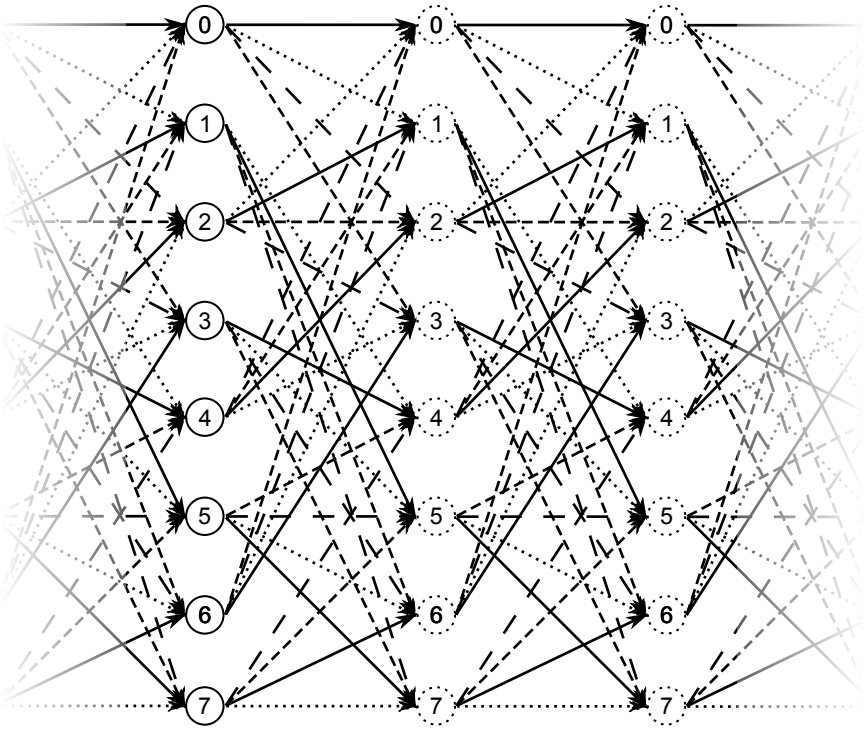
3.2 Représentation en treillis des codes QC-LDPC

Une structure de *treillis générique à deux états* a été présentée dans la section 3.1. Celle-ci s'adapte à la majorité des treillis des codes constituant les turbocodes courants. Dans cette section, nous réutilisons cette représentation pour définir différentes formes de treillis équivalents pour le décodage de codes QC-LDPC et nous établissons la notion d'entrelacement pour ces structures.

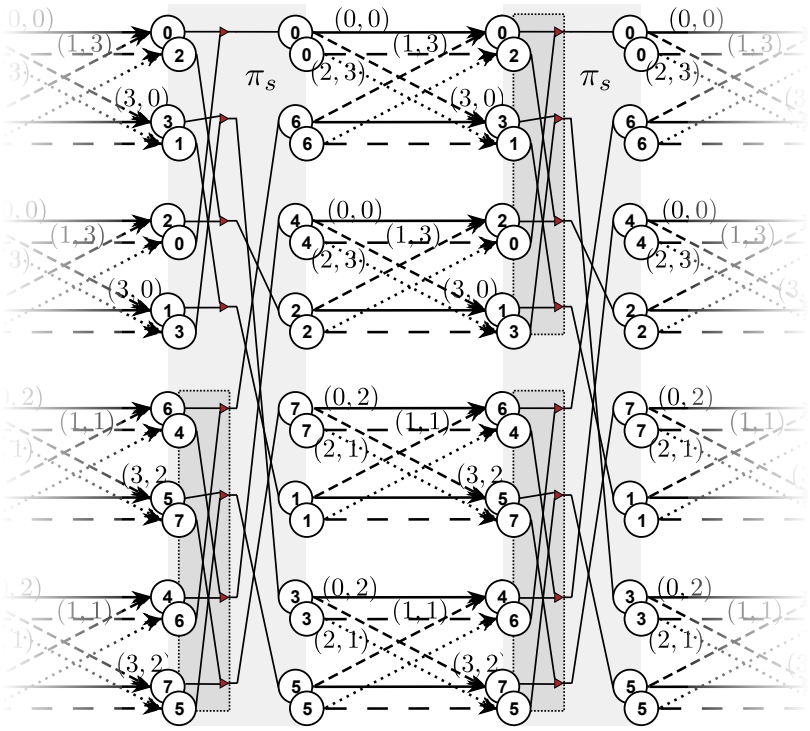
3.2.1 Transformation en treillis d'une équation de parité

Afin de décrire un code QC-LDPC sous la forme de treillis, il est nécessaire de caractériser le treillis associé à une équation de parité e_m fixée. Une équation de parité se traduit par l'opération (3.13).

$$\bigoplus_{n \in \mathcal{N}_m} c_n = 0 \tag{3.13}$$



(a)



(b)

FIGURE 3.3: Diagramme en treillis du code convolutif associé au standard HomePlug AV en représentation classique (a) et suivant une structure de treillis générique (b)

Les variables c_n impliquées dans l'équation sont indicées par l'ensemble $\mathcal{N}_m$ de cardinal $d_c^{\mathbf{m}}$. Il est possible de lister ces éléments. On pose $\pi_m(k)$ une fonction qui ordonne l'ensemble des indices $\mathcal{N}_m$. De ce fait, cette fonction vérifie (3.14).

$$\begin{array}{rccc} \pi_m : & [\![0; d_c^{\mathbf{m}}[\![& \rightarrow & \mathcal{N}_m \\ & k & \mapsto & n \end{array} \tag{3.14}$$

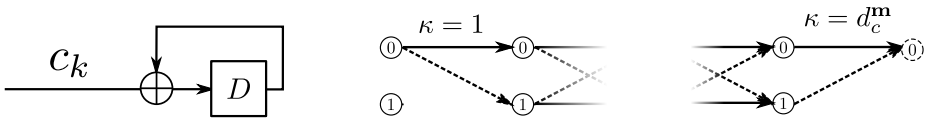
Dans ce cas, les opérations (3.13) et (3.15) sont équivalentes.

$$\bigoplus_{k=0}^{d_c^{\mathbf{m}}-1} c_{\pi_m(k)} = 0 \tag{3.15}$$

On propose alors d'ordonner les étapes de calcul de l'équation de parité associée. En appliquant un principe de récurrence, on pose une suite de $d_c^{\mathbf{m}}$ éléments notés S_k. Cette suite vérifie alors le système (3.16).

$$\left\{ \begin{array}{ccccc} S_0 & = & 0 & & \\ S_{k+1} & = & S_k + c_{\pi_m(k)} & & \\ S_{d_c^{\mathbf{m}}} & = & S_0 & = & 0 \end{array} \right. \tag{3.16}$$

Une équation de parité est caractérisée comme un code de type *Repeat-Acumulate* (RA) dont le diagramme est représenté sur la Figure 3.4a. Ces codes sont représentés en treillis à deux états. On dénomme alors celui-ci *treillis de contrainte de parité*. La Figure 3.4b fournie la représentation en treillis d'une équation de parité.



(a) (b)

Figure 3.4: Représentation d'un code *Repeat Accumulate* (a) et treillis associé à une équation de parité (b)

Une équation de parité est donc représentée comme un treillis à deux états, la fonction π_m (3.14) permet d'associer les indices de variables aux sections de ce treillis. Cette fonction est bijective et sa fonction réciproque notée π_m^{-1} définit dans ce cas la fonction d'entrelacement.

3.2.2 Transformation en treillis d'un groupe d'équations indépendantes

Le *treillis de contrainte de parité* est applicable sur tout type d'équation de parité, et ainsi sur tout code LDPC. Les propriétés architecturales des codes LDPC structurés comme les codes QC-LDPC permettent de faciliter le parallélisme de

décodage, mais également de définir une structure en treillis adaptative. Les codes QC-LDPC forment une concaténation parallèle de sous-codes $\mathcal{C}^{\mathbf{m}}$ regroupant les équations de parité dont l'indice est contenu dans l'ensemble $[\![m.z;(m+1).z[\![$. $\mathcal{C}^{\mathbf{m}}$ est un ensemble de z équations indépendantes de même degré de parité $d_c^{\mathbf{m}}$. En vertu de la relation explicitée dans la section 3.2.1, il existe une fonction bijective π_m reliant les indices n des variables c_n de chaque équation e_m et leur place dans un treillis à deux états k. Nous proposons de concaténer horizontalement l'ensemble de ces treillis. La fonction $\pi^{\mathbf{m}}$ désigne alors la relation entre la position d'une variable c_n sur le treillis équivalent noté alors *treillis d'ensemble de contraintes indépendantes*. Cette fonction est caractérisée par la relation (3.17).

$$\begin{array}{rccc} \pi^{\mathbf{m}}: & [\![0; z.d_c^{\mathbf{m}}[\![& \rightarrow & \bigcup_{\zeta=0}^{z-1}\mathcal{N}_{m.z+\zeta} \\ & \kappa & \mapsto & \pi^{\mathbf{m}}(\kappa) \end{array} \tag{3.17}$$

L'indépendance des équations de $\mathcal{C}^{\mathbf{m}}$ assure que les ensembles d'indices $\mathcal{N}_{m.z+\zeta}$ sont disjoints. La réunion des ensembles des indices de variables est de cardinal $z.d_c^{\mathbf{m}}$.

La fonction $\pi^{\mathbf{m}}$ s'exprime à l'aide de la fonction $\pi_{m.z}$. Cette dernière est utilisée comme fonction de référence pour caractériser l'ensemble du sous-code $\mathcal{C}^{\mathbf{m}}$. D'après les propriétés des matrices identités circulantes, chaque équation e_ζ de $\mathcal{C}^{\mathbf{m}}$ permet d'établir la relation (3.18).

$$\pi_\zeta(k) = \left\lfloor \frac{\pi_{m.z}(k)}{z} \right\rfloor .z + \Big((\pi_{m.z}(k))_{[z]} + \zeta \Big)_{[z]} \tag{3.18}$$

La fonction d'entrelacement associée vérifie alors (3.19).

$$\pi_\zeta^{-1}(n) = \pi_{m.z}^{-1}\left(\left\lfloor \frac{n}{z} \right\rfloor + (n-\zeta)_{[z]} \right) \tag{3.19}$$

Chaque équation de parité est concaténée en série. La position des variables est fournie par l'indice κ. À partir des éléments précédents, la fonction $\pi^{\mathbf{m}}$ se caractérise par (3.20).

$$\pi^{\mathbf{m}}(\kappa) = (\pi_{m.z}(\kappa_{[d_c^{\mathbf{m}}]})|z).z + \Big(\pi_{m.z}(\kappa_{[d_c^{\mathbf{m}}]})_{[z]} + (\kappa|d_c^{\mathbf{m}}) \Big)_{[z]} \tag{3.20}$$

Cette fonction est inversible et la fonction réciproque associée $\pi^{\mathbf{m}-1}$ (3.21) représente la fonction d'entrelacement associée au sous-code $\mathcal{C}^{\mathbf{m}}$.

$$\pi^{\mathbf{m}-1}(n) = \pi^{-1}_{(\kappa)_{[d_c^{\mathbf{m}}]}}(n) \tag{3.21}$$

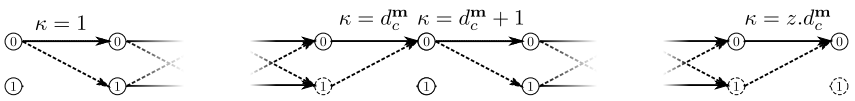


FIGURE 3.5: Représentation d'un *treillis d'ensemble de contraintes indépendantes* lié au sous-code $\mathcal{C}^{\mathbf{m}}$

La fonction $\pi^{\mathbf{m}}$ permet de définir un *treillis d'ensemble de contraintes indépendantes* lié au sous-code $\mathcal{C}^{\mathbf{m}}$ correspondant à la mise bout à bout des *treillis de contrainte de parité*. La FIGURE 3.5 présente les éléments de cette section et le diagramme obtenu.

La fonction $\pi^{\mathbf{m}}(\kappa)$ vérifie la propriété (3.22).

$$\pi^{\mathbf{m}}\left(\kappa + d_c^{\mathbf{m}}\right) = \begin{cases} \pi^{\mathbf{m}}(\kappa) + 1 & si \quad \left(\pi_0(\kappa_{[d_c^{\mathbf{m}}]})\right)_{[z]} + \left\lfloor \frac{\kappa}{d_c^{\mathbf{m}}} \right\rfloor + 1 - z < 0 \\ \pi^{\mathbf{m}}(\kappa) + 1 - z & sinon \end{cases} \tag{3.22}$$

Cette propriété rend possible le calcul itératif de la fonction d'entrelacement associée.

3.2.3 Transformation en treillis à 2.b états d'un groupe d'équations indépendantes

La section 3.2.2 propose la mise en série des treillis de contraintes lié au sous-code $\mathcal{C}^{\mathbf{m}}$. Cette représentation permet alors de définir une longueur pour cette représentation de $z.d_c^{\mathbf{m}}$ sections. Les équations de ce sous-groupe sont indépendantes. De ce fait, chaque équation peut être traitée en parallèle par une architecture de décodage. Dans cette section, on propose de traiter en parallèle $\mathbf{b}$ équations de parité, avec $\mathbf{b}$ une valeur entière comprise entre 1 et z. Afin de visualiser ce parallélisme, on propose de regrouper horizontalement les *treillis de contrainte de parité* en vérifiant l'unicité des chemins d'accès et en repérant par b l'indice du treillis vertical associé. À chaque indice de variable n correspond alors un emplacement horizontal κ ainsi qu'un indice vertical b. Le *treillis d'ensemble de contraintes indépendantes* présente alors la concaténation de $\lceil z/\mathbf{b} \rceil$ *treillis de contraintes multiples*. Dans cette section, on se propose de caractériser la fonction d'entrelacement associée à cette représentation.

La fonction $\pi_{\mathbf{b}}^{\mathbf{m}}(\kappa, b)$ fournie la relation entre un indice de variable n et son emplacement horizontal κ et son indice vertical b. Cette fonction se caractérise à partir de la fonction $\pi^{\mathbf{m}}$. Elle vérifie (3.23). Elle est bijective et sa fonction réciproque est notée $\pi_{\mathbf{b}}^{\mathbf{m}-1}(n)$.

$$\pi_{\mathbf{b}}^{\mathbf{m}}(\kappa, b) = \pi^{\mathbf{m}}\left((\kappa)_{[d_c^{\mathbf{m}}]} + \left\lfloor \frac{\kappa}{\mathbf{b}} \right\rfloor + b.d_c^{\mathbf{m}}\right) \tag{3.23}$$

La fonction $\pi_{\mathbf{b}}^{\mathbf{m}}(\kappa, b)$ permet de représenter le *treillis d'ensemble de contraintes indépendantes* comme la concaténation de *treillis de contraintes multiples* dont on note $\mathbf{b}$ le facteur de contrainte. La FIGURE 3.6 représente le diagramme obtenu. Il correspond à un treillis de $\left\lfloor \frac{z}{\mathbf{b}} \right\rfloor .d_c^{\mathbf{m}}$ sections.

3.2.4 Représentation d'un code QC-LDPC sous forme de treillis

La section 3.2.3 a mis en évidence la possibilité d'obtenir une fonction d'entrelacement spécifique à un sous-groupe d'équations indépendantes $\mathcal{C}^{\mathbf{m}}$ tout en exploitant

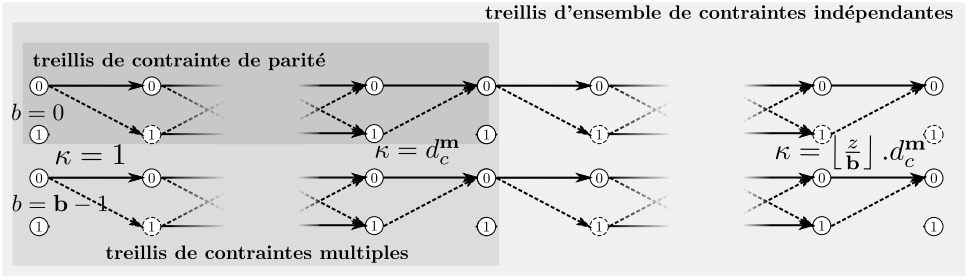


FIGURE 3.6: Représentation d'un *treillis d'ensemble de contraintes indépendantes* avec un facteur de contrainte **b**

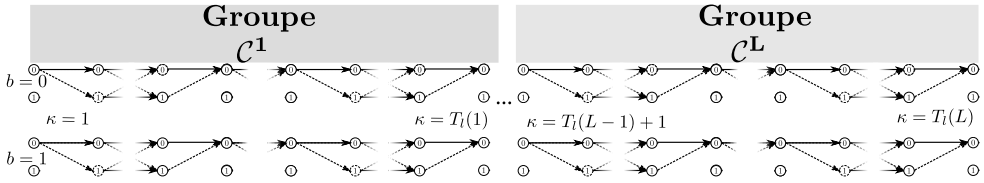


FIGURE 3.7: Représentation du *treillis d'itération* d'un code QC-LDPC

un degré de parallélisme **b** paramétrable. Dans cette section, on se propose d'obtenir une forme de treillis permettant de mettre à jour l'ensemble des équations de parité d'une matrice QC-LDPC. Celui-ci est nommé *treillis d'itération*. Pour cela, on définit une fonction d'adressage Π qui à un couple (κ, b) associe un indice de variable n. La FIGURE 3.7 illustre ce treillis pour un code QC-LDPC. Dans cette étude, la valeur de l'indice de parallélisme vertical **b** n'est pas spécifiée.

On propose d'ordonner les treillis obtenus dans la section précédente dans l'ordre horizontal, ce qui implique d'opérer les jonctions des sous-groupes d'équations indépendantes $\mathcal{C}^{\mathbf{m}}$ dans l'ordre croissant. Avant d'établir cette étape, on propose de noter $T_l^{\mathbf{m}}$ la longueur du *treillis d'ensemble de contraintes indépendantes* associé au sous-code $\mathcal{C}^{\mathbf{m}}$. Cette longueur est définie par la relation (3.24).

$$T_l^{\mathbf{m}} \triangleq \left\lfloor \frac{z}{\mathbf{b}} \right\rfloor . d_c^{\mathbf{m}} \tag{3.24}$$

On définit de même la longueur des m premiers sous-groupes par la relation (3.25).

$$T_l(m) \triangleq \sum_{\mathbf{m}=1}^{m} T_l^{\mathbf{m}} \tag{3.25}$$

La fonction Π (3.26) détermine la relation entre une indice (κ, b) du *treillis d'itération* et les positions des variables c_n du mot de code.

$$\Pi(\kappa, b) = \pi_{\mathbf{b}}^{\arg\max_{\mathbf{m}}(\mathbf{T_l(m)}<\kappa)}(\kappa - \max_{m}(T_l(m) < \kappa), b) \tag{3.26}$$

Toutes les contraintes de parité associées à une matrice QC-LDPC sont représentées sur le *treillis d'itération*. Cette section a permis de définir une relation entre les

différentes variables c_n d'un mot de code QC-LDPC et leur position sur un treillis à partir des paramètres de la matrice de parité du code. La longueur de celui-ci, noté T_{It}, varie en fonction du facteur contrainte $\mathbf{b}$ et du code QC-LDPC. La TABLE 3.1 fournit les longueur de ces treillis pour les matrices du standard IEEE 802.11n en fonction du facteur de contrainte $\mathbf{b}$. Elle dépend alors principalement des degrés de parité de la matrice de parité associée au code. Pour des matrices ayant un nombre de nœuds $\mathbf{d}^H$ du même ordre, les *treillis d'itération* sont du même ordre de grandeur. Cette propriété est notamment vérifiée pour les matrices du standard IEEE 802.11n.

N	R	z	T_{It} **pour** $\mathbf{b}=1$	T_{It} **pour** $\mathbf{b}=4$	T_{It} **pour** $\mathbf{b}=8$
648	1/2	27	2376	616	352
	2/3		2376	616	352
	3/4		2376	616	352
	5/6		2376	616	352
1296	1/2	54	4644	1204	602
	2/3		4752	1232	616
	3/4		4752	1232	616
	5/6		4590	1190	595
1944	1/2	81	6966	1806	968
	2/3		7128	1848	968
	3/4		6885	1785	935
	5/6		6399	1659	869

TABLE 3.1: Taille du *treillis d'itération* pour les matrices du standard IEEE 802.11n en fonction du facteur de contrainte $\mathbf{b}$

Cependant, chaque variable d'un code QC-LDPC est impliquée dans le calcul de plusieurs équations de parité. De ce fait, la fonction d'adressage Π n'est pas bijective, puisqu'à un indice n correspond exactement $d_v^{\mathbf{n}}$ couples distincts.

3.2.5 Transformation en treillis pour une matrice QC-LDPC

La fonction d'entrelacement associée à une matrice QC-LDPC dépend du facteur d'expansion z, du facteur de contrainte $\mathbf{b}$ ainsi que des indices de permutation $j_{\mathbf{m},\mathbf{n}}$ associés à un sous-code $\mathcal{C}^{\mathbf{m}}$ et à un groupe de variables compris dans l'ensemble $[\![\mathbf{n}.z;(\mathbf{n}+1).z[\![$. Afin d'illustrer l'entrelacement d'un code QC-LDPC, on propose d'étudier la matrice de parité associée au standard IEEE 802.11n de taille $N=648$ et de rendement $R=1/2$. Pour cette matrice, $z=27$.

Pour cela, on étudie l'ordonnancement associé au groupe $\mathcal{C}^{\mathbf{m}}$ pour $\mathbf{m}=0$. Les éléments de l'ensemble $\mathcal{N}_m$ sont listés dans l'ordre croissant. Ainsi, la fonction (3.14) définie dans la section 3.2.1 vérifie le système (3.27)

$$\begin{cases} \pi_0(0) = 0.z + j_{\mathbf{0,0}} = 0 \\ \pi_0(1) = 4.z + j_{\mathbf{0,4}} = 4.z + 0 \\ \pi_0(2) = 5.z + j_{\mathbf{0,5}} = 5.z + 0 \\ \pi_0(3) = 8.z + j_{\mathbf{0,8}} = 8.z + 0 \\ \pi_0(4) = 11.z + j_{\mathbf{0,11}} = 11.z + 0 \\ \pi_0(5) = 12.z + j_{\mathbf{0,12}} = 12.z + 1 \\ \pi_0(6) = 13.z + j_{\mathbf{0,13}} = 13.z + 0 \end{cases} \tag{3.27}$$

À partir de ce choix de liste, chaque fonction d'entrelacement $\pi_\zeta(k)$ définie par (3.18) vérifie le système (3.28).

$\forall \zeta \in [\![0.z; 1.z[\![$

$$\begin{cases} \pi_\zeta(0) = 0.z + (j_{\mathbf{0,0}} + \zeta)_{[z]} = \zeta \\ \pi_\zeta(1) = 4.z + (j_{\mathbf{0,4}} + \zeta)_{[z]} = 4.z + \zeta \\ \pi_\zeta(2) = 5.z + (j_{\mathbf{0,5}} + \zeta)_{[z]} = 5.z + \zeta \\ \pi_\zeta(3) = 8.z + (j_{\mathbf{0,8}} + \zeta)_{[z]} = 8.z + \zeta \\ \pi_\zeta(4) = 11.z + (j_{\mathbf{0,11}} + \zeta)_{[z]} = 11.z + \zeta \\ \pi_\zeta(5) = 12.z + (j_{\mathbf{0,12}} + \zeta)_{[z]} = 12.z + (1 + \zeta)_{[z]} \\ \pi_\zeta(6) = 13.z + (j_{\mathbf{0,13}} + \zeta)_{[z]} = 13.z + \zeta \end{cases} \tag{3.28}$$

On pose alors le facteur de contrainte $\mathbf{b} = 4$. Dans ce cas, les correspondances entre les indices de treillis (κ, b) et les indices du mot de code n sont obtenues en regroupant les caractéristiques de la relation (3.28) en fonction des ζ adjacents. Ainsi, avec $z = 27$, l'ensemble du sous-code constitue $\lceil z/b \rceil = 7$ *treillis de contraintes multiples*. Le *treillis d'ensemble de contraintes indépendantes* de ce sous-code est constitué de $7.d_c^{\mathbf{0}} = 49$ sections.

La fonction $\Pi(\kappa, b)$ vérifie sur cette section de treillis la relation (3.29) en posant $\zeta = \left\lfloor \frac{\kappa}{d_c^{\mathbf{0}}} \right\rfloor$.

$$\Pi(\kappa, b) = \pi_{4.\zeta+b}((\kappa)_{[d_c^{\mathbf{0}}]}) \tag{3.29}$$

Les premières sections du *treillis d'ensemble de contraintes indépendantes* sont illustrées sur la FIGURE 3.8. Le principe de transformation doit être appliqué pour tous les sous-codes $\mathcal{C}^{\mathbf{m}}$. On obtient alors le *treillis d'itération* lié à cette matrice.

3.3 Ordonnancement des mises à jour des treillis

Les sections 3.1 et 3.2 présentent respectivement une structure de treillis adaptée aux turbocodes et aux codes QC-LDPC pour une application de l'algorithme BCJR. Cette section étudie l'ordonnancement des calculs BCJR et représente le décodage de chaque code sur une itération de décodage. Nous analysons également la découpe de ces treillis pour améliorer le parallélisme de l'architecture tout en maîtrisant son impact sur les performances de décodage.

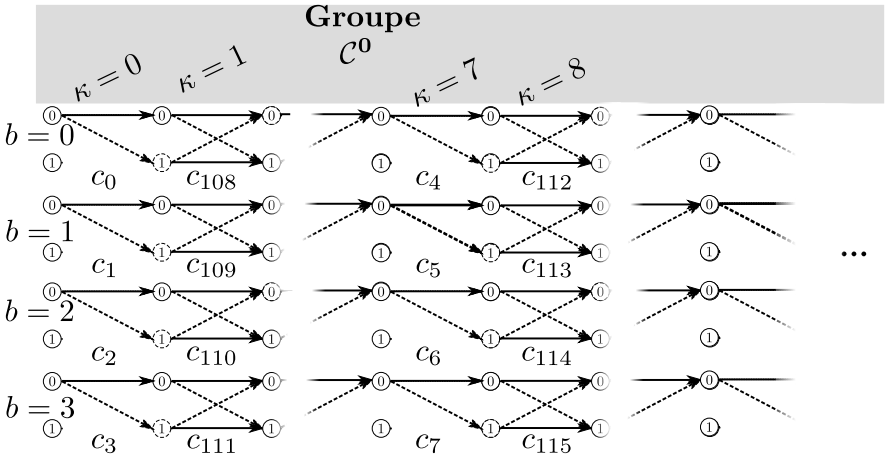


FIGURE 3.8: Représentation du *treillis d'itération* équivalent au code QC-LDPC du standard IEEE 802.11n ($R = 1/2$, $N = 648$)

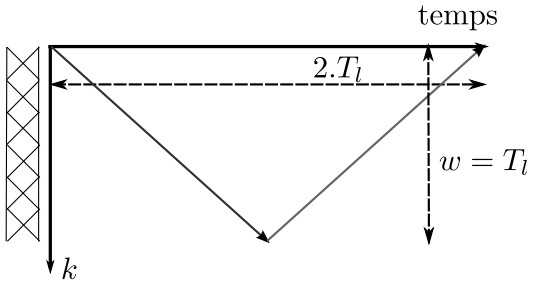
3.3.1 Ordonnancement des calculs BCJR

Le décodage BCJR appelle le calcul itératif des métriques cumulées $\alpha_k(s)$ et $\beta_k(s)$ qui représentent la probabilité que la machine de Markov soit à l'état s à la section k en connaissant l'ensemble des éléments passés et futurs du treillis. Ces opérations nécessitent donc de parcourir celui-ci suivant deux ordres *Aller* et *Retour*.

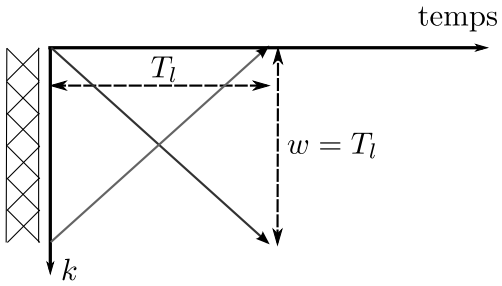
Dans un premier temps, on considère que le calcul de ces métriques est réalisé suivant une architecture de décodage complètement séquentielle en deux phases successives. Pour une architecture *Aller* puis *Retour* (ou *Forward-Backward*) [95], le calcul des métriques $\alpha_k(s)$ est réalisé en premier lieu sur l'ensemble du treillis. Celui des métriques $\beta_k(s)$ est effectué ensuite. La FIGURE 3.9a développe alors le parcours du treillis en fonction du temps. Lorsque cet ordre est inversé, on parle d'architecture *Retour* puis *Aller*. Ces ordonnancements imposent de stocker toutes les métriques cumulées sur la longueur du treillis T_l. Sur une architecture complètement séquencée, la latence de parcours du treillis est proportionnelle à $2.T_l$ cycles d'horloge.

Pour réduire cette latence de calcul, il est envisageable de calculer des métriques cumulées en parallèle. Ceci n'entraîne pas de dégradation de performances car les métriques $\alpha_k(s)$ et $\beta_k(s)$ sont indépendantes. La latence de calcul est divisée par 2. Pour une architecture séquentielle, la quantité de données stockées est équivalente à l'ordonnancement précédent. Cette architecture est représentée sur la FIGURE 3.9b. On la nomme généralement architecture en *Papillon*. [96] détaille l'architecture de décodage associée à ce type de décodage.

Le nombre de données stockées lors du décodage d'un code convolutif est proportionnel à la longueur du treillis. Pour éviter de stocker une telle quantité de données, il est courant d'approcher les métriques $\alpha_k(s)$ et $\beta_k(s)$ sur des tronçons de treillis plutôt que sur le treillis global. Cette technique permet alors de définir une fenêtre de traitement dont la taille est notée w. Le fenêtrage d'un treillis permet ainsi de



(a)



(b)

FIGURE 3.9: Ordonnancement du calcul des métriques cumulées suivant un ordre *Retour* puis *Aller* (a) et un ordre en *Papillon* (b)

réduire la quantité d'informations nécessaires au calcul des métriques $\alpha_k(s)$ et $\beta_k(s)$. Sans étude de parallélisme, cette opération n'améliore pas la rapidité de décodage du treillis. Cependant, le décodage avec cette technique est moins performant. En effet, en bordure de fenêtre, aucune connaissance sur les états n'est connue. Les décisions sur les éléments situés en bordure de fenêtre sont donc dégradés.

Pour réduire ces approximations, [97] introduit une phase d'entraînement de T_e sections sur les états précédents et suivants de la fenêtre du treillis pour un code convolutif. Sur cette partie de treillis, les métriques cumulées ne sont pas stockées mais elles sont nécessaires pour approcher plus efficacement les métriques $\alpha_k(s)$ et $\beta_k(s)$ en bordure de fenêtre. Le transfert des métriques entre fenêtres permet également d'améliorer ces approximations. La FIGURE 3.10 montre cet ordonnancement avec une *fenêtre glissante* sur le treillis d'un code circulaire. [98] reprend cette approche dans le cadre de décodage de turbocodes.

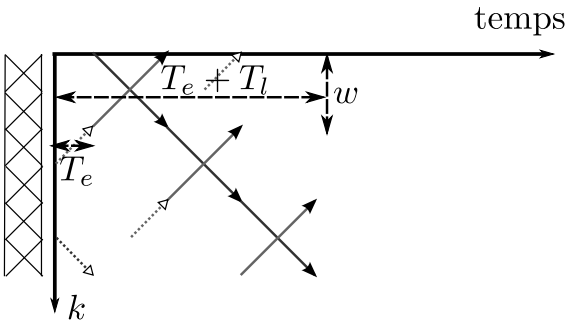


FIGURE 3.10: Ordonnancement du calcul des métriques cumulées suivant un ordre en *Papillon* avec fenêtre glissante de taille w et phase d'entraînement de longueur T_e

L'arrivée du décodage itératif a permis le développement d'une nouvelle possibilité d'apprentissage appelée NII [99] (*Next Iteration Initialization*) qui consiste à stocker les métriques cumulées en bordure des treillis entre les itérations afin de les réutiliser lors du décodage de l'itération suivante. Ce type de mise à jour permet alors de réduire les effets de bords. La FIGURE 3.11 représente dans ce cas le transfert des métriques $\alpha(k)$ et $\beta(k)$ entre chaque itération de décodage turbocode. La FIGURE 3.12 montre l'impact de la taille de fenêtre avec mise à jour NII sur un turbocode binaire de taille $K = 256$ et $R = 1/3$. Ces résultats sont obtenus en appliquant le décodage LogMAP en virgule flottante avec la technique de mise à jour des métriques cumulées NII pour 8 itérations. Pour ce décodage, aucune phase d'entraînement n'est prise en compte. Le transfert de métrique permet de limiter la dégradation du taux d'erreur binaire impacté par le fenêtrage du treillis.

Le traitement en fenêtres d'un treillis permet de paralléliser les étapes de calculs des métriques cumulées. De ce fait, il est possible d'améliorer les débits de décodage au prix d'une réduction des performances BER. Cependant, pour effectuer ce parallélisme, les treillis doivent permettre d'accéder aux données sans conflit d'accès.

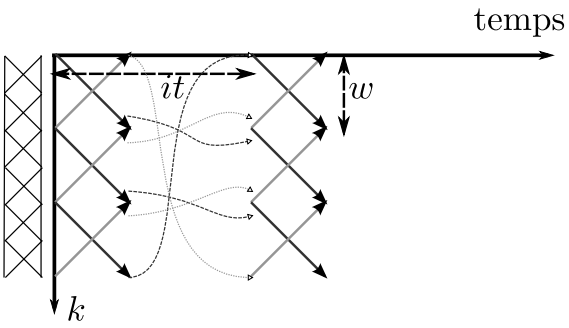


FIGURE 3.11: Calcul des métriques cumulées suivant un ordre avec application d'une fenêtre glissante et technique NII

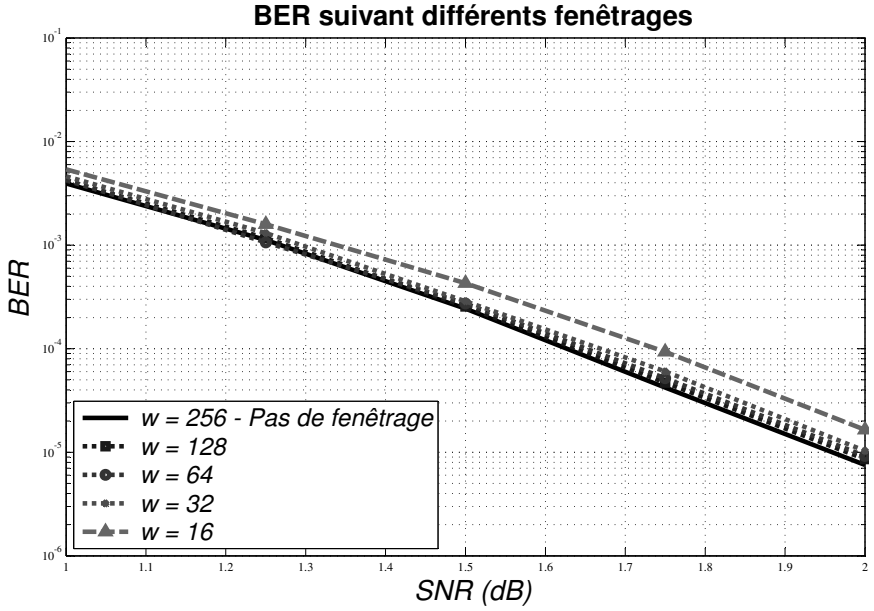


FIGURE 3.12: Impact de différents fenêtrages NII sur les performances de décodage d'un mot de code du standard 3GPP LTE ($K = 256$, $R = 1/3$)

Une discussion sur ce sujet est menée dans la section suivante.

D'autres ordonnancements ont été évalués pour répondre à des critères matériels. Les mémoires de thèse de O. MULLER [100] et D. GNAEDIG [101] font état de certains d'entre eux.

3.3.2 Tronçons de treillis

Travailler sur un tronçon du treillis permet d'améliorer le parallélisme de décodage et de diviser d'autant la latence de calcul du décodage total. Cependant, paralléliser des étapes de décodage augmente le nombre de requêtes en lecture et en écriture dans une mémoire. Ces requêtes ne doivent pas entrer en conflit. La gestion de l'ordre de lecture et d'écriture doit être étudiée afin d'éviter des conflits d'accès aux données, comme la lecture à plusieurs adresses d'un même banc de données à un cycle d'horloge fixé ou l'accès en lecture et en écriture sur une même adresse au même instant. [102] propose de modifier le plan d'enregistrement des mémoires pour éviter ces conflits d'accès.

[103] propose de caractériser les entrelaceurs des turbocodes afin d'éviter ces conflits d'accès. La gestion du nombre de tronçons P de longueur W est étudiée en fonction de l'entrelaceur du turbocode associé. [103] propose de caractériser un entrelaceur sans conflit d'accès par la définition 4.

Définition 4. *Un entrelaceur π est dit sans conflit d'accès pour une découpe en P tronçons de W sections de treillis si :*

- *P et W sont des diviseurs de la taille de treillis totale T_l*
- $\forall \zeta \in [\![0; W[\![\ \forall a, b \in [\![0; P[\![, a \neq b \Leftrightarrow$ *(3.30)*

$$\left\lfloor \frac{\pi(\zeta + a.W)}{W} \right\rfloor \neq \left\lfloor \frac{\pi(\zeta + b.W)}{W} \right\rfloor \tag{3.30}$$

Si un entrelaceur vérifie cette propriété, alors il est possible de scinder le treillis en P tronçons de W sections. Pour une valeur de section fixée k, à chaque indice de section k' vérifiant $k' \equiv (k)_{[W]}$ correspond un banc mémoire distinct.

[103] montre que les entrelacements QPP sont sans conflits d'accès pour tout diviseur de T_l. Ils sont alors dits *Maximum Contention-Free*. L'entrelaceur associé au code LTE vérifie cette propriété.

L'entrelaceur associé au turbocode du standard HomePlug AV vérifie (3.31).

$$\pi(j) = \left(\left(Q\left((j)_{[P_0]} \right) - P_0 \times (j|P_0) \right) \right)_{[T_l]} \tag{3.31}$$

La fonction $Q(x)$ est périodique de période P_0, ce qui permet de vérifier la propriété (3.32).

$\forall \zeta \in [\![0; W[\![\ \forall a, b \in [\![0; P[\![, a \neq b$

$$\left\lfloor \frac{\pi(\zeta + a.W)}{W} \right\rfloor = \left\lfloor \frac{Q(\zeta)}{W} + P - a \right\rfloor \neq \left\lfloor \frac{\pi(\zeta + b.W)}{W} \right\rfloor \tag{3.32}$$

Ceci permet de dire que l'entrelaceur est sans conflit pour $P = P_0/T_l$. Cette propriété permet d'établir le parallélisme de décodage en fonction des caractéristiques des mots de code. La TABLE 3.2 reprend cette étude dans le cas de turbocodes 3GPP LTE et HomePlug AV.

Standard	P **max**	**Conditions**
LTE	8	$K \leq 512$
	16	$512 \leq K < 1024$
	32	$1024 \leq K < 2048$
	64	$2048 \leq K$
HomePlug AV	8	$K = 128$ ou $K = 256$
	16	$K = 1088$
	52	$K = 4160$

TABLE 3.2: Nombre maximum de processeurs pour éviter les conflits d'accès suivant les standards 3GPP LTE et HomePlug AV

3.3.3 Représentation d'une itération de décodage

Les turbocodes convolutifs sont la concaténation en parallèle de deux codes convolutifs de même treillis et de même taille T_l. Dans la section 3.2, les codes QC-LDPC ont été transformés en une succession de treillis dont la taille est dépendante de la régularité des matrices. Afin de préserver les ressources matérielles de stockage, la section 3.3.2 a permis d'étudier les caractéristiques des codes convolutifs afin de paramétrer un fenêtrage des treillis tout en évitant les conflits d'accès mémoire.

Le décodage des turbocodes consiste à décoder successivement deux codes convolutifs correspondant au décodage de l'ordre naturel et au décodage de l'ordre entrelacé. Un cycle de décodage entrelacé puis naturel représente une itération de décodage. Pour représenter le décodage turbocode sur une itération, nous proposons de joindre bout à bout ces deux treillis et de dédier des processeurs de décodage BCJR en fonction des tronçons de treillis présentés précédemment. Une itération de décodage représente dans ce cas $T_{It} = 2.T_l$ sections de treillis. Avec la technique de tronçonnage, chaque processeur décode une série de T_l/P par code convolutif et par itération. Chaque processeur opère ainsi le décodage de son tronçon de treillis suivant l'ordre de son choix. Ce choix d'ordre de traitement est représenté sur la FIGURE 3.13a où le code convolutif est circulaire. L'occupation des processeurs de décodage est représentée en fonction du tronçon de treillis occupé et du déroulement du processus de décodage sur une itération.

Le décodage de turbocodes nécessite de parcourir deux treillis par itération. De ce fait, une itération de décodage représente une longueur équivalente de treillis de $T_{It} = 2.T_l$. En opérant un fenêtrage sur l'ordonnancement associé au turbocode

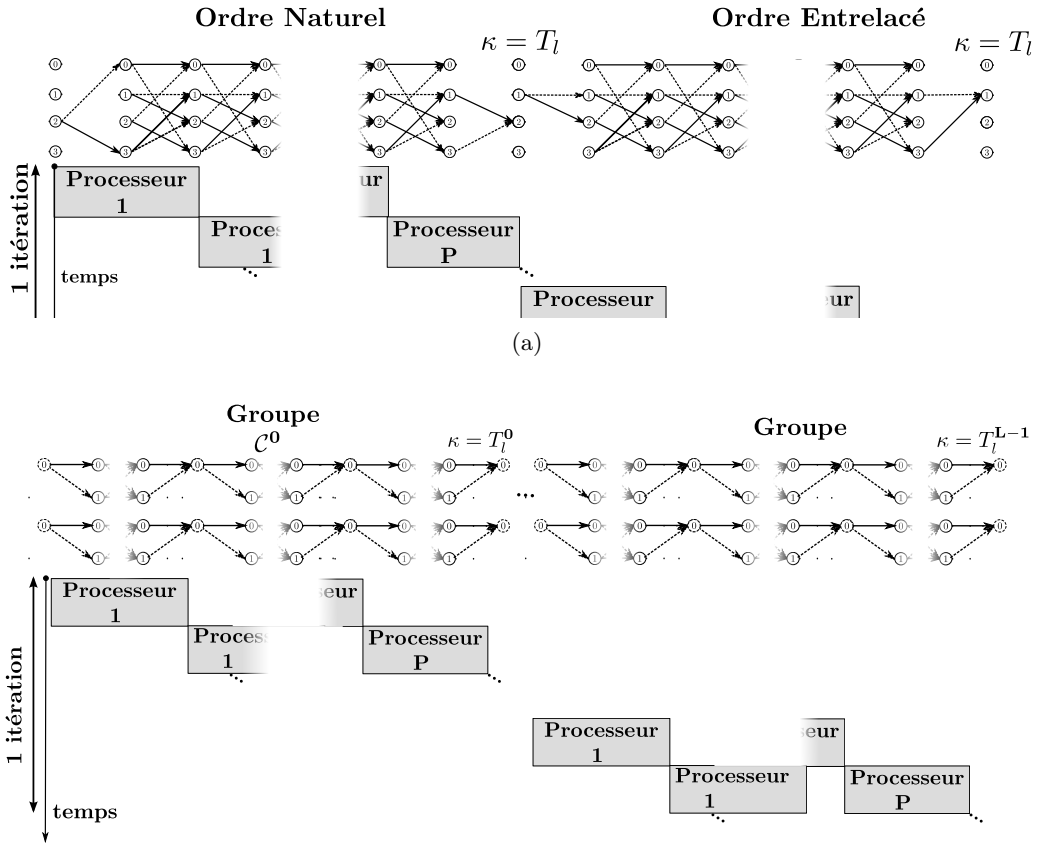


FIGURE 3.13: Ordonnancement des treillis pour turbocode (a) et pour codes QC-LDPC composé de L sous-groupes indépendants (b)

(sous réserve d'avoir un entrelaceur sans conflit d'accès), les treillis naturel et entrelacé se scindent en P tronçons de longueurs T_l/P. Chaque tronçon est mis à jour suivant des processeurs de calculs distincts, permettant ainsi de réduire la latence de décodage total. L'ordre de calcul interne suit un ordre de décodage en *Papillon* fenêtré en longueur w sans phase d'entraînement. Les treillis sont mis à jour en série, depuis l'ordre entrelacé vers l'ordre naturel. Cet ordre de calcul et les parallélismes sont soulignés sur la FIGURE 3.13a. D'autres ordres de décodage sont envisageables. [104] propose de décoder les treillis naturels et entrelacés en parallèle sans obtenir de performances acceptables. La FIGURE 3.13a représente les parcours des treillis d'un turbocodes à travers le temps. Sur cette figure, le décodage est effectué par P processeurs distincts ce qui permet de réduire la latence de décodage.

Le tronçonnage des *treillis d'itération* des codes QC-LDPC est défini dès la construction de ceux-ci. Chaque sous-codes $\mathcal{C}^{\mathbf{m}}$ représente un treillis de longueur variable dépendante du facteur de contrainte $\mathbf{b}$ et du degré de parité $d_c^{\mathbf{m}}$. Chacun de ces treillis est lui-même la concaténation en série de $\left\lceil \frac{z}{\mathbf{b}} \right\rceil$ *treillis de contraintes multiples* dont les mises à jour sont indépendantes. De ce fait, chaque *treillis d'ensemble de contraintes indépendantes* est subdivisé en $P = \left\lceil \frac{z}{\mathbf{b}} \right\rceil$ tronçons. L'indépendance des contraintes garantit alors la préservation des performances de décodage. Un ordonnancement des mises à jour par couches horizontales entraîne le traitement en série de chaque *treillis d'ensemble de contraintes indépendantes*. Chaque *treillis de contraintes multiples* respecte cependant les contraintes d'accès mémoire et peut ainsi être mis à jour en parallèle. De ce fait, le décodage d'une itération d'un code QC-LDPC s'effectue en parallèle pour chaque équation d'un sous-groupe indépendant et en série pour chaque *treillis d'ensemble de contraintes indépendantes*. La FIGURE 3.13b représente cet ordre de calcul pour plusieurs processeurs de décodage parallèles. Dans ce cas, les processeurs se répartissent le décodage des *treillis de contraintes multiples* d'un ensemble indépendant ce qui garantit que chaque processeur décode le même nombre de sections tout en évitant les conflits d'accès aux informations.

3.4 Transformation de l'ordonnancement appliqué aux codes QC-LDPC

Une architecture de décodage BCJR entraîne une latence de calcul non négligeable pour le décodage de codes QC-LDPC. Cette partie reprend l'étude réalisée dans la section 3.2 et modifie la structure du treillis pour réduire l'impact de cette latence sur les débits de décodage d'un code QC-LDPC.

3.4.1 Intérêt de l'approche

Le chapitre 2 présente l'algorithme BCJR comme un algorithme de décodage permettant de décoder aussi bien les turbocodes que les codes QC-LDPC. Les premières parties de ce chapitre ont permis de détailler une structure en treillis représentée

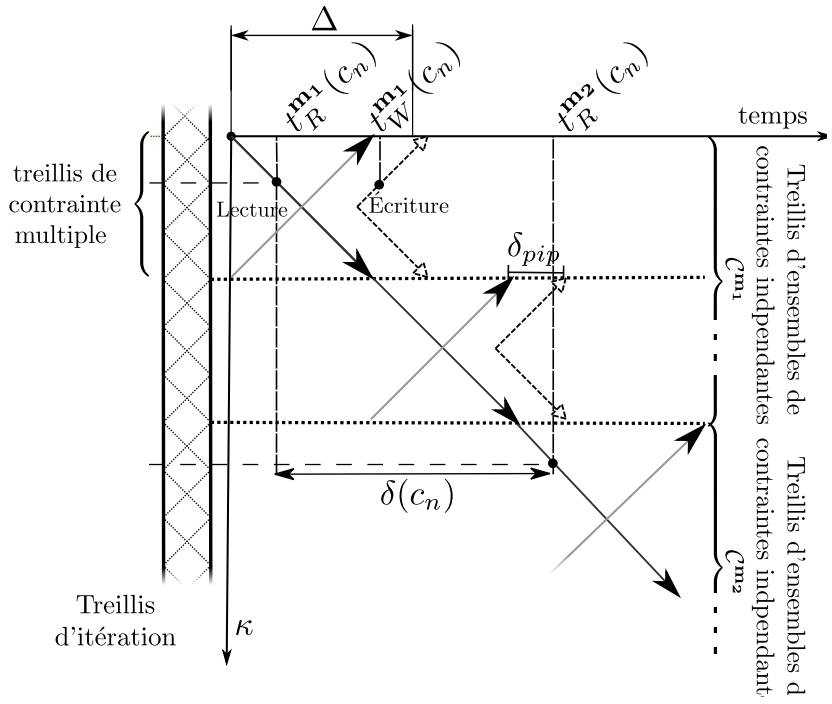


FIGURE 3.14: Chronogramme du décodage en *Papillon* du treillis d'un code QC-LDPC

sur une itération de décodage. De ce fait, une structure de décodage conjointe doit permettre de décoder l'ensemble de ces treillis.

Cependant, la section 2.3 du chapitre 2 montre que le calcul des informations liées à une section de treillis demande un délai de propagation élevé. Sur une architecture séquentielle, les étapes de calculs sont synchronisées afin de réduire le chemin critique de l'opération au prix d'un temps de latence δ_{pip} exprimé en cycles d'horloge. Une telle architecture de décodage sera présentée dans le chapitre 4 (section 4.1). Nous y montrons que cette opération est de l'ordre de 8 cycles d'horloge.

Le traitement en treillis nécessite une lecture de celui-ci dans les sens *Aller* et *Retour*. Le délai entre la lecture d'une donnée $t_R^{\mathbf{m}}(c_n)$ et l'écriture des informations mises à jour $t_W^{\mathbf{m}}(c_n)$ dépend alors de l'emplacement k de la variable c_n dans le *treillis de contraintes multiples*, de son emplacement dans le *treillis d'itération* κ, du choix de parcours du treillis, ainsi que de la latence de calcul δ_{pip}.

Lors du décodage de codes QC-LDPC, les données de décision *a posteriori* sont réinjectées au cours de l'itération de décodage. La description des *treillis d'ensemble de contraintes indépendantes* garantit que chaque variable c_n n'est utilisée qu'au plus une fois. Dans ce cas, la mise à jour de cette variable n'est pas requise sur la durée totale du décodage de ces treillis. Cette condition n'est plus vérifiée entre deux sous-codes indépendants $\mathcal{C}^{\mathbf{m_1}}$ et $\mathcal{C}^{\mathbf{m_2}}$. De ce fait, certaines données calculées pour une variable c_n au cycle d'horloge $t_W^{\mathbf{m_1}}(c_n)$ sont réutilisées à un temps d'accès $t_R^{\mathbf{m_2}}(c_n)$. Ce cas d'usage est illustré sur le chronogramme de la FIGURE 3.14 pour un

ordonnancement en *Papillon*. Ce chronogramme met en valeur le délai de réutilisation $\delta(c_n)$ entre l'utilisation d'une variable c_n pour deux sous-codes adjacents. Pour une architecture en *Papillon*, ce délai respecte la relation (3.33).

$$\delta(c_n) = t_R^{\mathbf{m_2}}(c_n) - t_A^{\mathbf{m_1}}(c_n) \tag{3.33}$$

De même, le temps de traitement du *treillis de contraintes multiples* noté Δ défini par la relation (3.34) pour une architecture de décodage en *Papillon* est exploité dans la suite de cette section.

$$\Delta = d_c^{\mathbf{m}} + \delta_{pip} \tag{3.34}$$

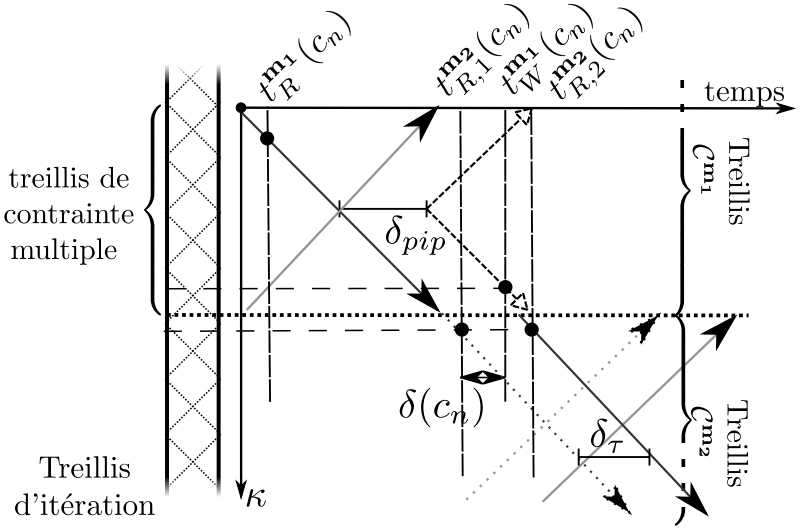
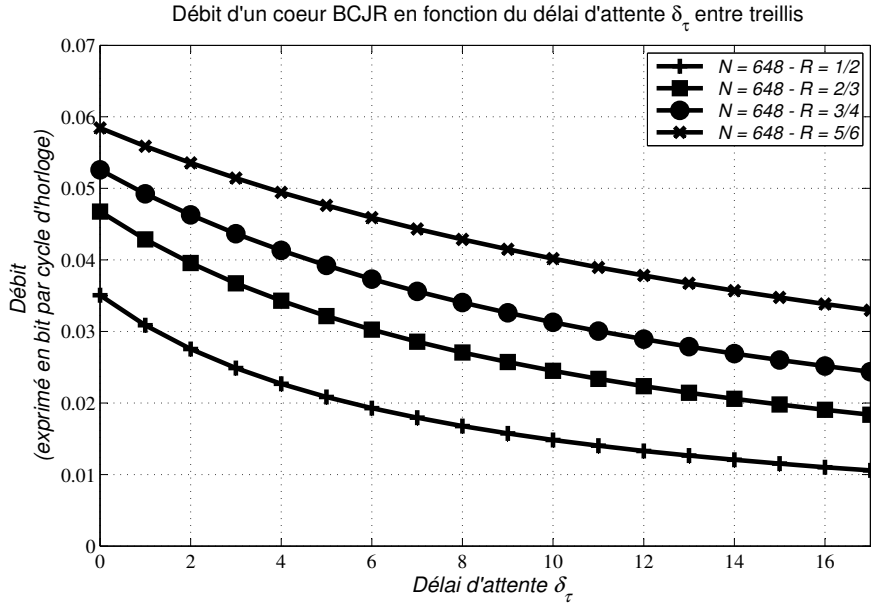


FIGURE 3.15: Conflit entre une donnée mise à jour et son accès dans le cas du décodage en *Papillon* du treillis d'un code QC-LDPC. On y représente le conflit en traits continus et sa résolution en traits pleins

La différence entre le temps d'écriture de la donnée précédemment mise à jour et sa lecture dans le groupe suivant doit rester positive pour chaque cas de transition de *treillis d'ensemble de contraintes indépendantes*. La FIGURE 3.15 schématise un cas où l'accès est requis avant que la mise à jour ne soit effective. Alors qu'une architecture de cœur BCJR doit permettre la transition des treillis sans délai pour garantir des débits compétitifs, ces cas d'accès obligent à casser le treillis et à arrêter le processus de décodage pour vider les pipelines et attendre la disponibilité de la donnée mise à jour. On note alors δ_τ le délai d'attente minimal pour éviter ces cas d'accès. De ce fait, le décodeur est mis en attente de δ_τ cycles d'horloge entre chaque *treillis d'ensemble de contraintes indépendantes*. Ce délai d'attente vérifie alors (3.35).

$$\delta_\tau = -\min_{\mathbf{m_2}>\mathbf{m_1}} \left(t_R^{\mathbf{m_2}}(c_n) - t_A^{\mathbf{m_1}}(c_n), 0\right) \tag{3.35}$$



(a)

FIGURE 3.16: Représentation de la dégradation des débits due au délai d'attente δ_τ entre *treillis d'ensemble de contraintes indépendantes* sur les codes QC-LDPC du standard IEEE 802.11n

On suppose des architectures de décodage à un processeur BCJR avec un parcours de treillis en *Papillon* qui décode des MCS du standard IEEE 802.11n. La FIGURE 3.16 montre l'impact du délai d'attente sur les débits des codes de ce standard. Ce débit D, exprimé ici en bits systématiques par cycle d'horloge, est calculé en prenant en compte le nombre de treillis N_t par itération de décodage et le nombre d'itérations N_{it} de décodage, en fonction de la relation (3.36).

$$D = \frac{K}{N_{it} \times \sum_{n=1}^{N_t} (w(n) + \delta_\tau)} \tag{3.36}$$

Les codes QC-LDPC sont composés d'une multitudes de *treillis de contraintes multiples*. Pour obtenir ces résultats, nous avons posé le facteur de contrainte à $\mathbf{b} = 4$. Le nombre d'itérations a été fixé à 15 pour ces codes comme pour [91]. Le débit de décodage est impacté par le rendement des codes QC-LDPC. En effet, bien que chaque code bénéficie d'un nombre de sections de même ordre en vertu de la TABLE 3.1, les degrés de parité et donc les tailles de fenêtre sont plus importantes sur les codes de fort rendement. Les codes à rendement élevé sont constitués de moins de *treillis d'ensemble de contraintes indépendantes* par itération de décodage. Enfin, le décodage d'un code de rendement élevé amène plus d'informations décodées, ce qui explique l'allure des courbes de la FIGURE 3.16.

Pour améliorer les débits, il est possible de ne pas réinjecter l'information mise à jour pour certaines variables c_n dans le processus de décodage. En remplaçant cette

information par une valeur neutre $\mathrm{L}^{a}_{\mathbf{m_2}}(c_n) = 0$, l'ensemble des variables impactées par l'équation suivante n'est pas mis à jour du fait de la faible complexité du treillis à deux états. En injectant la donnée $\mathrm{L}^{a}_{\mathbf{m_2}}(c_n) = \mathrm{L}^{a}_{\mathbf{m_1}}(c_n)$, ceci permet de réduire les dégradations engendrées.

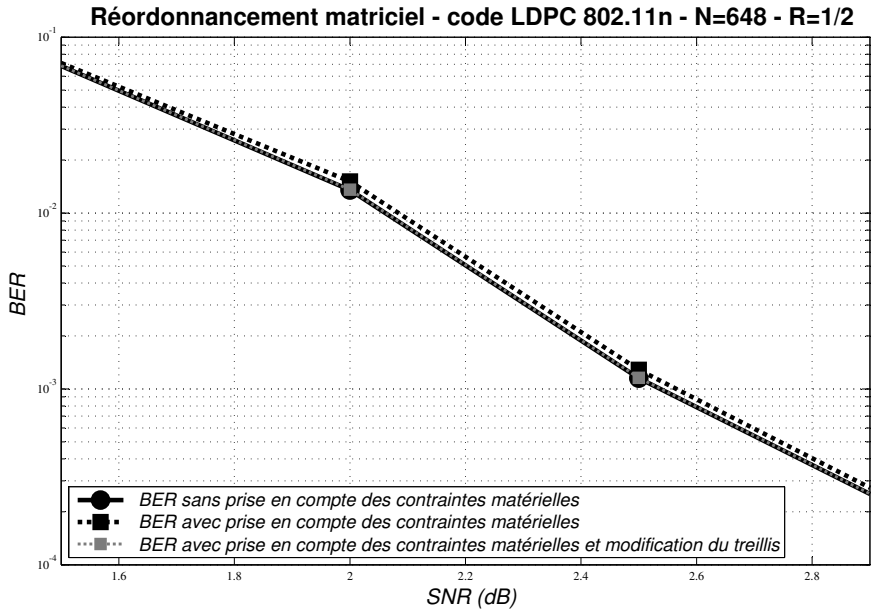


FIGURE 3.17: Performance de décodage suivant les contraintes matérielles pour $\delta_{pip} = 8$

Nous supposons une architecture de décodage qui évolue suivant l'ordonnancement en *Papillon*. Pour cette structure, nous supposons que le délai inter treillis δ_τ soit nul. Dans ce cas, un délai de calcul BCJR δ_{pip} nul permet d'obtenir une performance de décodage sans prendre en compte les contraintes matérielles. Cependant, dans la pratique, cette latence n'est pas nulle. Lorsque ce délai est de 8 cycles d'horloge, les performances sont dégradées. Celles-ci sont représentées sur la FIGURE 3.17 pour le décodage de la matrice IEEE 802.11n ($N = 648$, $R = 1/2$) pour une mise à jour Max-LogMAP sans échelonnage. Dans cette simulation, l'information manquante est compensée par l'information précédemment acquise sur la variable. Le non remplacement de cette information entraîne une dégradation plus élevée encore puisque l'ensemble des informations du treillis ne sont pas mises à jour dans ce cas. Pour récupérer des performances équivalentes entre les deux cas d'usage, il suffit de fixer δ_τ en fonction de δ_{pip}.

Dans cette section, nous envisageons d'étudier l'ordre des opérations permettant de réduire le délai d'attente δ_τ entre les *treillis d'ensemble de contraintes indépendantes* pour garantir une amélioration des débits tout en conservant les performances originales. Nous reviendrons exploiter les dernières courbes de la FIGURE 3.17. Ce chapitre à fait l'objet d'une publication [105].

3.4.2 Étude des délais du treillis

Afin d'éviter les pertes de performance de décodage, le délai d'attente de δ_τ cycles d'horloge est nécessaire pour éviter tout conflit d'accès mémoire. Cette section propose d'étudier les conditions que doit remplir le treillis d'un code QC-LDPC afin d'éviter les conflits d'accès aux informations mises à jour.

Une variable c_n n'est utilisée qu'au maximum une fois par *treillis d'ensemble de contraintes indépendantes* lié au sous-code $\mathcal{C}^{\mathbf{m}}$ à un emplacement κ défini dans la section 3.2. Pour effectuer la mise à jour de l'équation de parité, la dernière information de cette variable doit être accessible à l'instant $t_R^{\mathbf{m}}(c_n)$ dépendant de l'ordonnancement du calcul des métriques cumulées et de l'emplacement κ. Le temps d'écriture de la mise à jour résultante $t_W^{\mathbf{m}}(c_n)$ dépend de l'ordonnancement du treillis ainsi que de l'emplacement de la variable dans celui-ci. Le délai entre le temps de lecture et le temps d'écriture vérifie (3.37) pour un ordonnancement en *Papillon*.

$$t_W^{\mathbf{m}}(c_n) - t_R^{\mathbf{m}}(c_n) = 2.\left|\left\lceil \frac{d_c^{\mathbf{m}}}{2} \right\rceil - k\right| + \delta_{pip} \tag{3.37}$$

Ce délai n'est pas uniformément réparti. En effet, les éléments situés au centre du *treillis de contraintes multiples* seront mis à jour sur moins de cycles d'horloge que les éléments les plus excentrés. Cette notion est représentée sur la FIGURE 3.18 l'ordonnancement en *Papillon*.

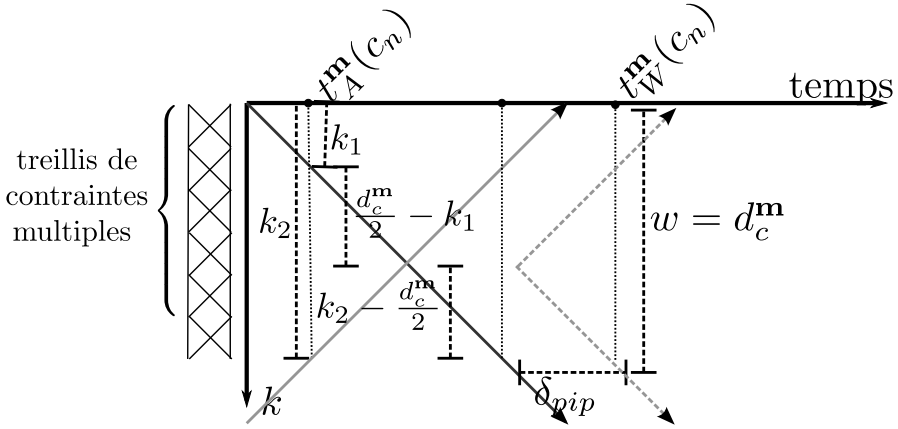


FIGURE 3.18: Représentation des délais pour un ordonnancement des calculs du treillis en *Papillon*

Ce délai est majoré par le temps de traitement du *treillis de contraintes multiples* Δ défini par (3.34) pour un ordonnancement en *Papillon*. On obtient donc un encadrement du temps de calcul néccessaire à chaque variable par la relation (3.38).

$$\delta_{pip} \leq t_W^{\mathbf{m}}(c_n) - t_R^{\mathbf{m}}(c_n) \leq \Delta \tag{3.38}$$

Si la variable c_n interagit avec des équations situées dans deux groupes indépendants $\mathcal{C}^{\mathbf{m_1}}$ et $\mathcal{C}^{\mathbf{m_2}}$, on définit $\delta(c_n)$ comme le délai entre chaque accès en lecture des

données relatives à cette variable. $\delta(c_n)$ est définie par la relation (3.33).

De (3.38) ou déduit alors une condition suffisante (3.39) sur $\delta(c_n)$.

$$\delta(c_n) = t_R^{\mathbf{m_2}}(c_n) - t_R^{\mathbf{m_1}}(c_n) < \Delta \quad \Rightarrow \quad t_R^{\mathbf{m_2}}(c_n) > t_W^{\mathbf{m_1}}(c_n) \tag{3.39}$$

Cette relation assure de la disponibilité de la dernière mise à jour pour toute variable c_n dont le délai de réutilisation est supérieur à la latence Δ. À l'inverse, pour garantir que toutes les mises à jour sont réutilisées par le processus de décodage, il faut imposer une latence δ_τ vérifiant (3.40).

$$\delta_\tau > \Delta - \min(\delta(c_n)) \tag{3.40}$$

3.4.3 Augmentation du délai de réutilisation d'une information mise à jour

Le délai de réutilisation $\delta(c_n)$ dépend du parallélisme du cœur de décodage $\mathbf{b}$. Afin d'optimiser les débits sans risquer de dégrader les performances de décodage, il est nécessaire d'arranger l'ordre de calcul des mises à jour.

On définit alors V_c l'ensemble des variables communes à deux groupes indépendants successifs $\mathcal{C}^{\mathbf{m}}$ et $\mathcal{C}^{\mathbf{m+1}}$. Pour chaque variable c_n de V_c, il existe deux équations distinctes m_1 et m_2 utilisant cette variable. On cherche alors à définir le délai de réutilisation $\delta(c_n)$ entre le délai de premier et de second accès à la variable.

On décrit l'ensemble des valeurs de la fonction δ sur l'ensemble V_c par l'ensemble Ω_v définit par (3.41). La valeur minimale de cet ensemble est alors notée m_Ω.

$$\Omega_v = \{\delta(c_n),\ c_n \in V_c\} \subset \mathbb{N}_+^* \tag{3.41}$$

On choisit alors de modifier l'ordre des opérations du second sous-groupe $\mathcal{C}^{\mathbf{m+1}}$ à l'aide d'une permutation s, de sorte que

$$\forall m_2 \in \mathcal{C}^{\mathbf{m+1}},\ \exists! m'_2 \in \mathcal{C}^{\mathbf{m+1}},\ m'_2 = s(m_2) \tag{3.42}$$

Il devient alors possible d'étudier le délai de réutilisation avec cette permutation. Ce délai permuté est noté $\delta^s(c_n)$. La notion de délai de réutilisation est étendue avec la relation (3.43).

$$\delta^s(c_n) = t_s^{\mathbf{m+1}}(c_n) + T^{\mathbf{m}} - t_R^{\mathbf{m}}(c_n) \tag{3.43}$$

avec

$$t_s^{\mathbf{m+1}}(c_n) = \left(\left\lfloor \frac{(s(m))_{[z]}}{\mathbf{b}} \right\rfloor . d_c^{\mathbf{m+1}} + \pi_{m_2}^{-1}(c_n) \right) \tag{3.44}$$

La valeur minimale de l'ensemble image obtenu Ω_v^S est alors notée m_Ω^S. Le délai de réutilisation maximal pour chaque équation est obtenu en modifiant l'ordre de décodage du groupe $\mathcal{C}^{\mathbf{m+1}}$ en choisissant la permutation maximisant le délai de réutilisation minimale m_Ω^S. En procédant de manière itérative sur chaque sous-groupe, on obtient ainsi une optimisation des délais de réutilisation.

Dans la pratique, le calcul de l'entrelacement d'une permutation aléatoire est complexe. Une permutation circulaire est un choix sous-optimal convenable. Il n'entraîne pas de surcoût du calcul de l'entrelacement, puisque la redéfinition du treillis consiste à transformer les indices de circulation sur des blocs lignes. De ce fait la matrice LDPC résultant de cette transformation reste une matrice QC-LDPC. Le calcul de l'entrelacement correspond donc toujours aux équations fournies dans la section 3.2.

0	-	-	-	0	0	-	-	0	-	-	0	1	0	-	-	-	-	-	-	-	-	-
22	17	-	-	0	-	0	0	12	-	-	-	-	0	0	-	-	-	-	-	-	-	-
6	-	0	-	10	-	-	-	24	-	0	-	-	-	0	0	-	-	-	-	-	-	-
2	-	-	0	20	-	-	-	25	0	-	-	-	-	-	0	0	-	-	-	-	-	-
23	-	-	-	3	-	-	-	0	-	9	11	-	-	-	-	0	0	-	-	-	-	-
24	-	23	1	17	-	3	-	10	-	-	-	-	-	-	-	-	0	0	-	-	-	-
25	-	-	-	8	-	-	-	7	18	-	-	0	-	-	-	-	-	-	0	0	-	-
13	24	-	-	0	-	8	-	6	-	-	-	-	-	-	-	-	-	-	-	0	0	-
7	20	-	16	22	10	-	-	23	-	-	-	-	-	-	-	-	-	-	-	0	0	-
11	-	-	-	19	-	-	-	13	-	3	17	-	-	-	-	-	-	-	-	0	0	-
25	-	8	-	23	18	-	14	9	-	-	-	-	-	-	-	-	-	-	-	-	0	0
3	-	-	-	16	-	-	2	25	5	-	-	1	-	-	-	-	-	-	-	-	-	0

(a)

0	-	-	-	0	0	-	-	0	-	-	0	1	0	-	-	-	-	-	-	-	-	-
10	15	-	-	5	-	15	15	0	-	-	-	-	15	15	-	-	-	-	-	-	-	-
10	-	4	-	14	-	-	-	1	-	4	-	-	-	4	4	-	-	-	-	-	-	-
10	-	-	8	1	-	-	-	6	8	-	-	-	-	-	8	8	-	-	-	-	-	-
10	-	-	-	17	-	-	-	14	-	23	25	-	-	-	-	14	14	-	-	-	-	-
11	-	10	15	4	-	17	-	24	-	-	-	-	-	-	-	-	14	14	-	-	-	-
21	-	-	-	4	-	-	-	3	14	-	-	23	-	-	-	-	-	23	23	-	-	-
21	5	-	-	8	-	16	-	14	-	-	-	-	-	-	-	-	-	8	8	-	-	-
25	11	-	7	13	1	-	-	14	-	-	-	-	-	-	-	-	-	-	18	18	-	-
12	-	-	-	20	-	-	-	14	-	3	18	-	-	-	-	-	-	-	-	1	1	-
1	-	11	-	26	21	-	17	12	-	-	-	-	-	-	-	-	-	-	-	-	3	3
18	-	-	-	4	-	-	17	13	20	-	-	16	-	-	-	-	-	-	-	-	-	15

(b)

FIGURE 3.19: Représentation de la matrice LDPC classique (a), remodelage pour un traitement optimisé en couches horizontales (b)

L'application de cette méthode sur la matrice IEEE 802.11n de taille $N = 648$ et de rendement $R = 1/2$ permet de transformer la matrice de parité classique FIGURE 3.19a en une matrice aux délais de réutilisation optimisés FIGURE 3.19b par une opération sur les lignes. Cette matrice de parité ne modifie cependant pas le code QC-LDPC initial.

La FIGURE 3.20 indique le nombre de nœuds dont le calcul de la mise à jour est en conflit en fonction du délai δ_{pip} nécessaire pour vider les tuyaux. Il suppose que l'architecture de décodage évolue suivant l'ordonnancement en *Papillon* avec δ_τ fixé

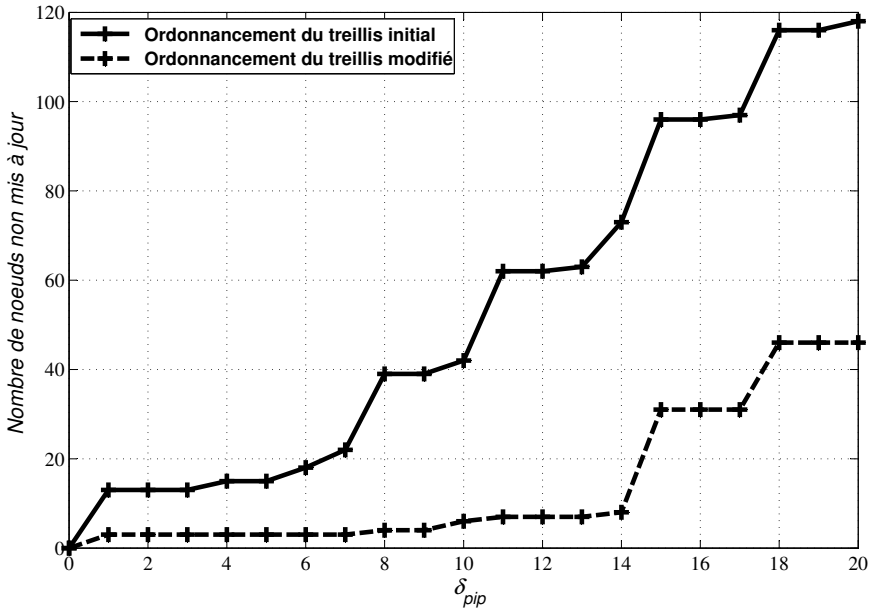


FIGURE 3.20: Impact du délai permettant de vider les pipelines δ_{pip} sur le nombre de nœuds non actualisés en fonction du treillis initial et du treillis modifié

à 0. La longueur du pipeline permet dans ce cas d'évaluer le nombre de mises à jour en conflit.

Dans le chapitre 4, la latence δ_{pip} est évaluée à 8 cycles d'horloge. Dans ce cas, l'ordonnancement classique oblige à éviter 39 mises à jour afin de satisfaire à une architecture aux débits optimisés. Avec l'ordonnancement modifié, seules 2 mises à jour ne sont plus satisfaites. La FIGURE 3.17 montre dans ce cas les dégradations de performances entre ces deux entrelacements de décodage.

3.5 Conclusion

Dans ce chapitre, la structure des codes convolutifs associés aux codes constituant les turbocodes a été détaillée. Une représentation de ces treillis suivant des transitions à deux états généralisés est rendu possible pour la plupart des turbocodes binaires. Cette structure s'adapte également aux turbocodes double-binaires. Les conditions d'une telle représentation ont été soulignées dans une première section de chapitre. Le cas particulier du treillis associé au turbocode des standards LTE fait l'objet d'une publication dans [106]. Cette représentation permet entre autre de mutualiser les ressources matérielles de décodage pour une architecture de décodage flexible.

Une seconde partie de ce chapitre est dédiée à la transformation des codes QC-LDPC en treillis indépendants. Afin de mutualiser au mieux les ressources matérielles dans le cas d'un décodeur conjoint, ces treillis ont été regroupés suivant des

contraintes de taille et d'indépendance des variables. Cette structure de treillis fait intervenir un facteur de contrainte multiple **b** dont la valeur doit être étudiée pour une convergence de décodage entre un code QC-LDPC et un turbocode. Ce paramètre permet également de caractériser la complexité de décodage d'une itération QC-LDPC en fonction d'un nombre de sections de treillis équivalent. Ce facteur est également exploité dans le chapitre 4 pour justifier un choix de parallélisme de décodage.

Dans un troisième temps, l'ordonnancement des mises à jour du treillis est discuté. Les sens de calcul *Aller -Retour* et *Papillon* sont exploitables sur tout type de treillis. La caractéristique des entrelaceurs turbocodes est discutée afin de définir une taille de fenêtre adaptée entre décodage turbocode et décodage QC-LDPC. Le faible nombre de sections associées à ces dernières ne permettent pas d'appliquer le transfert de métrique entre itérations NII. Cependant, un ordre de calcul sur une itération a été mis en évidence pour chaque type de code.

Le dernier point de ce chapitre amène une notion de délai de calcul pour le décodage de codes QC-LDPC. Ce délai est peu pris en compte pour le décodage de turbocode du fait de son faible impact sur les débits de décodage. Cependant, la multitude de treillis pour représenter un code QC-LDPC oblige à retravailler l'ordonnancement des calculs afin d'augmenter les débits de décodage tout en limitant l'impact sur les performances en termes de taux d'erreurs. Cette section a fait l'objet d'une publication dans [105].

Au final, ce chapitre souligne l'intérêt de nombreux paramètres qui permettent de structurer une architecture de décodage conjointe et flexible. Pour un algorithme BCJR, les fonctions d'entrelacement ont été établies pour chaque code. L'ordonnancement des opérations nécessaires aux calculs des métriques d'un treillis suit les représentations génériques en treillis à deux états et l'ordonnancement des opérations a été retravaillé pour améliorer les performances de décodage en termes de débits et en termes de taux d'erreur.

Chapitre 4

De l'architecture à l'intégration sur FPGA et ASIC d'un décodeur générique et flexible

Ce chapitre applique les concepts présentés dans les chapitres précédents pour la conception de plusieurs architectures de décodage. Le chapitre 2 a démontré la pertinence de l'application d'un algorithme de décodage BCJR pour le décodage conjoint de codes QC-LDPC et de turbocodes. La structure en treillis présentée dans le chapitre 3 est réutilisée dans la première partie de ce chapitre pour proposer un cœur de processeur BCJR commun à plusieurs standards. Les propriétés de reconfiguration sont également détaillées pour un cas de convergence du turbocode binaire et un code QC-LDPC. Ce cœur reconfigurable est ensuite intégré dans plusieurs architectures adaptées au décodage itératif appliqué aux codes correcteurs d'erreurs des standards 3GPP LTE et IEEE 802.11n. Les caractéristiques d'entrelacement et les contraintes matérielles sont détaillées pour ce cas d'usage. L'étude de parallélisme réalisée dans le chapitre 3 est également reprise pour proposer une architecture de décodage aux débits améliorés. Les choix architecturaux pour résoudre les conflits d'accès aux mémoires et la flexibilité de décodage sont également expliqués. Ces architectures sont vérifiées en dernier lieu sur les cibles FPGA et ASIC afin de caractériser les propriétés matérielles pour cette convergence de décodage.

4.1 Conception du cœur SISO

Dans cette section, l'étude de représentation des treillis associés aux codes QC-LDPC et turbocodes est reprise pour la réalisation d'une architecture de décodage d'un cœur de processeur BCJR appelé cœur SISO (*Soft-In Soft-Out*) dans la suite. Une preuve de concept de cette architecture est également présentée pour répondre à un cas de convergence 3GPP LTE et IEEE 802.11n. La reconfiguration de cette architecture est également dimensionnée.

4.1.1 Interface du cœur de processeur SISO

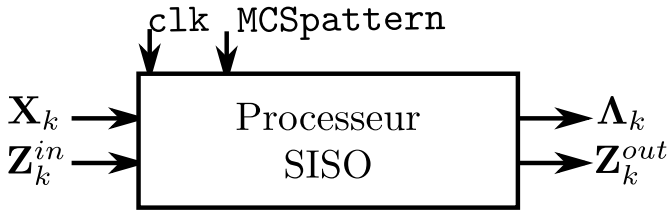
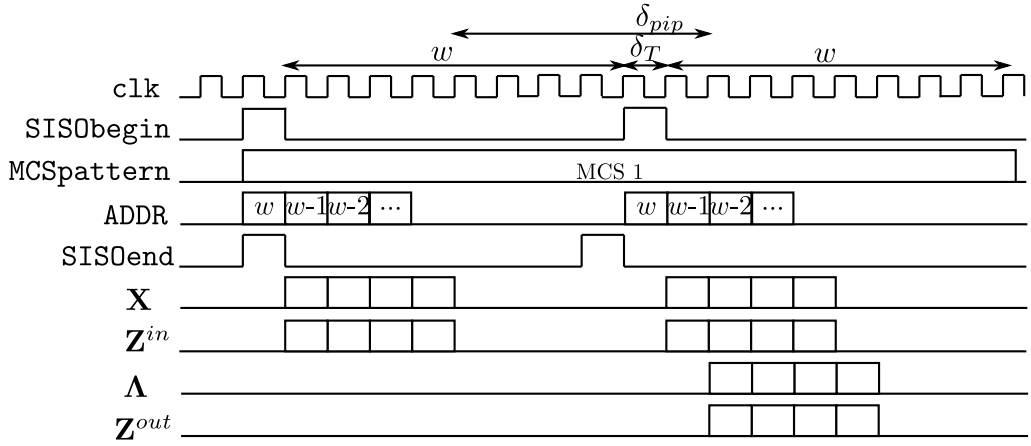


FIGURE 4.1: Le cœur de processeur SISO vu comme une boîte noire

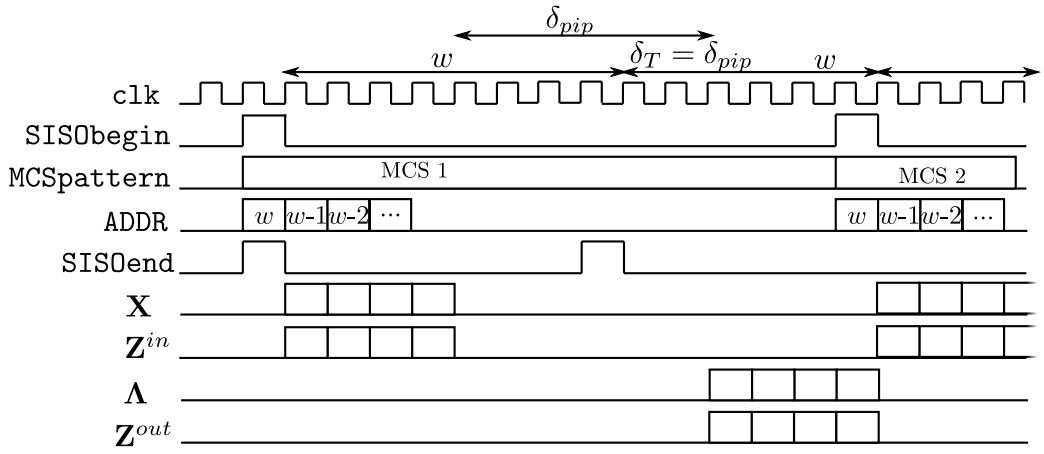
Le cœur de processeur SISO exécute toutes les opérations de calcul sur des treillis ou des tronçons qui ont été déterminés dans le chapitre 3. Ce processeur doit ordonner la lecture des informations du treillis utiles à la section k dans le sens croissant pour l'ordre *Aller* et décroissant pour l'ordre *Retour*.

L'interface du cœur de décodage SISO est présentée sur la FIGURE 4.1. Le cœur de processeur SISO doit respecter les critères de flexibilité de l'usage. Pour cela un jeton `MCSpattern` est ajouté à la structure pour modifier le contexte de décodage à la volée. La suite de ce chapitre montre que la flexibilité est engendrée par des choix de démultiplexage répartis sur les différents modules de décodage. À partir de vecteurs de données intrinsèques $\mathbf{X}_k$ et extrinsèques $\mathbf{Z}_k^{in}$ explicités dans la suite, ce processeur effectue les opérations de calcul des métriques de branche $\texttt{MBR}_k()$ puis des métriques cumulées $\texttt{MAC}_k(s)$ nécessaires à la mise à jour des vecteurs de données de décision $\mathbf{\Lambda}_k$ et de données extrinsèques $\mathbf{Z}_k^{out}$ du treillis.

Le cœur de décodage SISO est un élément qui est initialisé par l'extérieur au moyen d'un signal `SISObegin`. Lorsque ce signal est actif, le coeur SISO émet les adresses de lecture de données intrinsèques $\mathbf{X}_k$ et extrinsèques $\mathbf{Z}_k^{in}$. Lorsque la fenêtre est complètement parcourue, un signal `SISOend` indique la disponibilité du processeur pour le décodage d'un autre treillis. Les données de décision $\mathbf{\Lambda}_k$ et extrinsèques $\mathbf{Z}_k^{out}$ ne sont alors pas intégralement fournies par le décodeur. Un délai de δ_T cycles d'horloge permet alors la reconfiguration de ce cœur entre deux treillis décodés. La FIGURE 4.2a montre un chronogramme pour le décodage de deux treillis de même structure et la FIGURE 4.2b pour deux treillis de structures différentes.



(a)



(b)

FIGURE 4.2: Chronogramme de l'exploitation des interfaces du cœur SISO pour deux treillis de même structure (a) et pour deux treillis de structures différentes (b)

4.1.2 Architecture pour treillis générique

Les étapes de la mise à jour d'un treillis suivant l'algorithme BCJR sont représentées à travers trois modules : le module de calcul des métriques de branche **BMP**, le module de calcul des probabilités d'états **AMP** et le module de décision **DEC**. Des éléments de mémoires **SYST RAM** et **ACC RAM** permettent de stocker les métriques de branches et les métriques cumulées en fonction de leur usage dans ces différents blocs. Deux autres modules π_{BM} et π_s permettent d'orienter les différentes métriques en fonction des paramètres du treillis.

4.1.2.1 Gestion des ordonnancements du parcours du treillis

Le choix d'ordonnancement du décodage du treillis intervient dans l'architecture interne du cœur SISO.

Les ordonnancements du décodeur SISO ont été introduits dans le chapitre 3 section 3.3.1. Deux structures de décodage sont envisagées. Une première structure fonctionne sur un ordonnancement *Retour* puis *Aller* qui favorise une plus faible attribution de ressources matérielles. Une seconde préfère un ordonnancement en *Papillon* qui privilégie la réduction de la latence de traitement du cœur SISO.

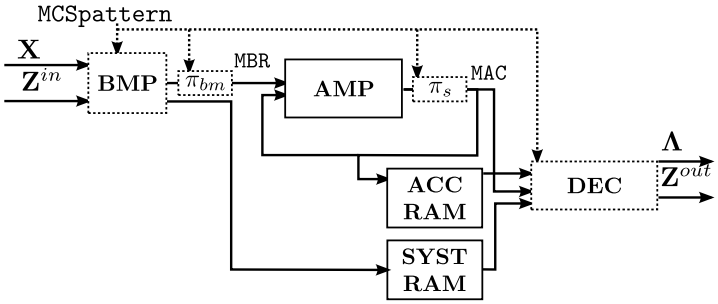
Pour un bon fonctionnement du cœur SISO et afin de limiter les conflits d'accès, les informations sur les sections de treillis ne sont lues qu'une fois quelque soit le choix de parcours du treillis. Elles sont ensuite stockées dans une mémoire LIFO notée **SYST RAM**. Le processeur SISO est divisé en fonction des modules **BMP**, **AMP** et **DEC**. La FIGURE 4.3 représente les liens entre les différents modules pour un ordre de calcul *Aller* puis *Retour* sur la FIGURE 4.3a et pour un ordre de calcul en *Papillon* sur la FIGURE 4.3b. L'agencement de ces modules est commun à tous les codes traités. Le cœur SISO conserve donc le même aspect pour tout type de convergence de standard.

Les modules sont cependant architecturés en fonction des scénarios de convergences envisagés. La section suivante revient sur ces architectures.

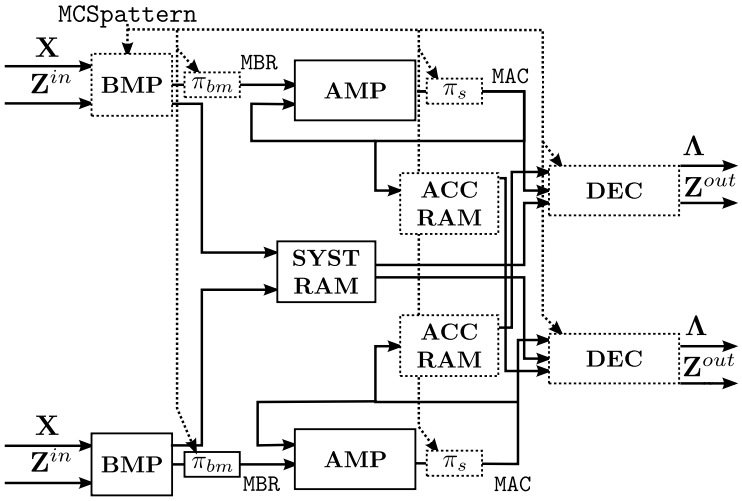
4.1.2.2 Module de calcul des métriques de branche (BMP)

Le module de calcul des métriques de branche **BMP** nécessite de connaître les informations du treillis de la section courante k. Du fait des caractéristiques de décodage itératif, les informations du treillis sont constituées de données intrinsèques et de données extrinsèques.

Les données intrinsèques correspondent aux informations systématiques (s_k^i suivant les standards) et aux informations redondantes (p_k^i) de la section courante k provenant du démodulateur pour le décodage des turbocodes. Un cœur SISO compatible avec le code convolutif 3GPP LTE exploite les Log-Rapports de Vraisemblance $\mathrm{L}^c(s_k)$ et $\mathrm{L}^c(p_k)$. Pour le décodage d'un code QC-LDPC, on privilégie la dernière mise à jour des $\mathbf{b}$ variables impliquées à la section κ du *treillis d'itération*, soit les informations $\mathrm{L}^a\left(c_{\Pi(\kappa,0)}\right)$, ... $\mathrm{L}^a\left(c_{\Pi(\kappa,\mathbf{b}-1)}\right)$. Dans ce cas, la fonction $\Pi(\kappa, b)$ suit la transformation en treillis établie dans le chapitre 3 (section 3.2.4). Le vecteur



(a)



(b)

FIGURE 4.3: Schéma de l'architecture présentant le cheminement des données dans le cas d'un cœur de processeur SISO en ordonnancement *Retour* puis *Aller* (a) et en ordonnancement en *Papillon* (b)

$\mathbf{X}_k$ reprend la structure de ce treillis. Pour un cas de convergence avec un facteur de contrainte $\mathbf{b}$ de 4, comme pour le cas d'une convergence 3GPP LTE et IEEE 802.11n, une section de treillis sollicite deux informations intrinsèques (3GPP LTE) ou quatre informations mises à jour (tout code QC-LDPC). De ce fait, ce vecteur $\mathbf{X}_k$ suit la relation (4.1).

$$\mathbf{X}_k \triangleq \begin{pmatrix} \mathbf{X}_k^{\mathbf{a}} \\ \mathbf{X}_k^{\mathbf{b}} \\ \mathbf{X}_k^{\mathbf{c}} \\ \mathbf{X}_k^{\mathbf{d}} \end{pmatrix} = \begin{pmatrix} \mathrm{L}^c\left(s_k\right) \\ \mathrm{L}^c\left(p_k\right) \\ 0 \\ 0 \end{pmatrix}_{\text{LTE}} = \begin{pmatrix} \mathrm{L}^a\left(c_{\Pi(\kappa,0)}\right) \\ \mathrm{L}^a\left(c_{\Pi(\kappa,1)}\right) \\ \mathrm{L}^a\left(c_{\Pi(\kappa,2)}\right) \\ \mathrm{L}^a\left(c_{\Pi(\kappa,3)}\right) \end{pmatrix}_{\text{QC-LDPC}} \tag{4.1}$$

Le vecteur d'informations extrinsèques $\mathbf{Z}_k^{in}$ contient les données échangées entre chaque décodeur dans le cas d'un décodage turbocode. Pour le cas QC-LDPC, seule l'information extrinsèque calculée par la même équation à l'itération précédente est intéressante. En effet, la dernière décision d'une information sur une variable contient à la fois l'information du canal et l'information extrinsèque du précédent décodage. Transférer uniquement cette donnée permet alors de réduire la quantité d'information nécessaire au décodage et de réduire la complexité du calcul des métriques de branches. De ce fait, le vecteur d'informations extrinsèques $\mathbf{Z}_k^{in}$ ne possède qu'une composante dans le cas d'un décodage 3GPP LTE et $\mathbf{b}$ composantes pour tout code QC-LDPC. Dans le cas d'un cœur SISO compatible avec les standards 3GPP LTE [22] et IEEE 802.11n [14], ce vecteur est défini par relation (4.2).

$$\mathbf{Z}_k^{in} \triangleq \begin{pmatrix} \mathbf{Z}_k^{in,\mathbf{a}} \\ \mathbf{Z}_k^{in,\mathbf{b}} \\ \mathbf{Z}_k^{in,\mathbf{c}} \\ \mathbf{Z}_k^{in,\mathbf{d}} \end{pmatrix} = \begin{pmatrix} \mathrm{L}^e\left(s_k\right) \\ 0 \\ 0 \\ 0 \end{pmatrix}_{\text{LTE}} = \begin{pmatrix} \mathrm{L}_m^e\left(c_{\Pi(\kappa,0)}\right) \\ \mathrm{L}_m^e\left(c_{\Pi(\kappa,1)}\right) \\ \mathrm{L}_m^e\left(c_{\Pi(\kappa,2)}\right) \\ \mathrm{L}_m^e\left(c_{\Pi(\kappa,3)}\right) \end{pmatrix}_{\text{QC-LDPC}} \tag{4.2}$$

La FIGURE 4.4 propose une architecture avec mutualisation des ressources pour le calcul des métriques de branche associées à ce type de convergence de standard. Les ressources affectées conjointement aux traitements des treillis turbocode et QC-LDPC sont représentées par des traits pleins, tandis que les ressources non mutualisées sont tracées en traits pointillés (cas QC-LDPC) ou en traits mixtes (cas turbocode). Cette structure permet de faire coïncider les éléments de calculs pour ces deux cas de décodage en évitant le gaspillage de ressources par la multiplication des usages et en limitant les choix de parcours entre les cas d'usages. Ainsi, sur cette structure, le jeton `MCSpattern` intervient uniquement sur la sélection de certaines sorties opératoires afin de permettre un changement de contexte à la volée.

4.1.2.3 Module de calcul des probabilités d'états (AMP)

Le module de calcul des probabilités d'états **AMP** utilise les informations $\texttt{MBR}_k(i)$ obtenues du précédent module pour définir les métriques cumulées $\texttt{MAC}_k(s)$. Pour ce module, la technique CSA [85] introduite dans la section 2.3.2 est appliquée selon

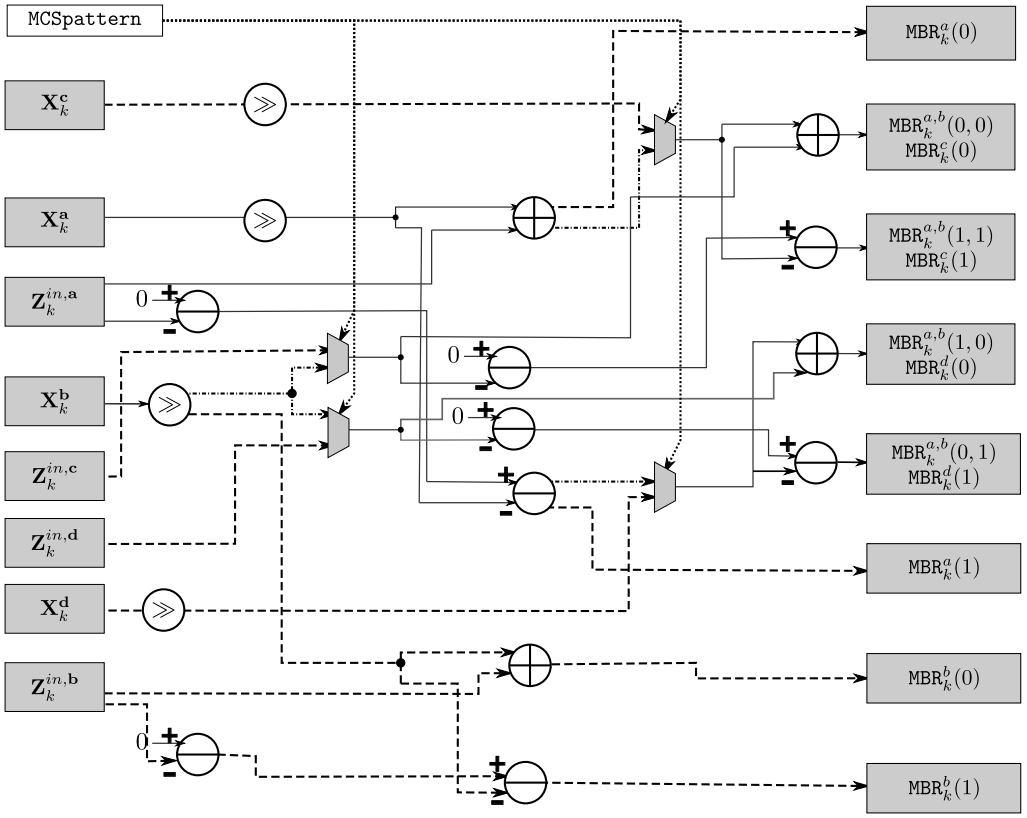


FIGURE 4.4: Mutualisation des ressources matérielles pour le calcul des métriques de branche du module **BMP** dans un cas de convergence 3GPP LTE et IEEE 802.11n

la structure de treillis générique introduite au long du chapitre 3. Ceci permet de réduire le chemin critique du circuit et mutualise les routes de transitions de messages grâce à la représentation des sections de treillis générique à deux états (section 3.1.1).

La FIGURE 4.5 présente la mise en parallèle du calcul des métriques cumulées pour une architecture binaire à 8 états en fonction du cheminement du treillis. La structure de l'accumulateur à 2 états suit l'architecture CSA présentée dans le chapitre 2 et détaillée sur la FIGURE 2.14b. Cette étape impacte le chemin critique de cette architecture et réduit la fréquence d'utilisation du cœur de processeur pour ce type d'architecture.

Chaque élément de calcul à deux états suit la structure d'une *section de treillis générique à deux états* introduite dans le chapitre 3 (section 3.1.1). Ces blocs de calcul sont donc inchangés pour tout contexte de décodage. Le jeton `MCSpattern` permet d'éviter la seconde étape de calcul dans un contexte de convergence entre un turbocode double-binaire et un code QC-LDPC. Cependant, les métriques de branches $\mathtt{MBR}_k(i)$ doivent être routées en fonction des étiquettes de branches définies par le treillis du code équivalent proposé dans le chapitre 3. Ce choix de routage

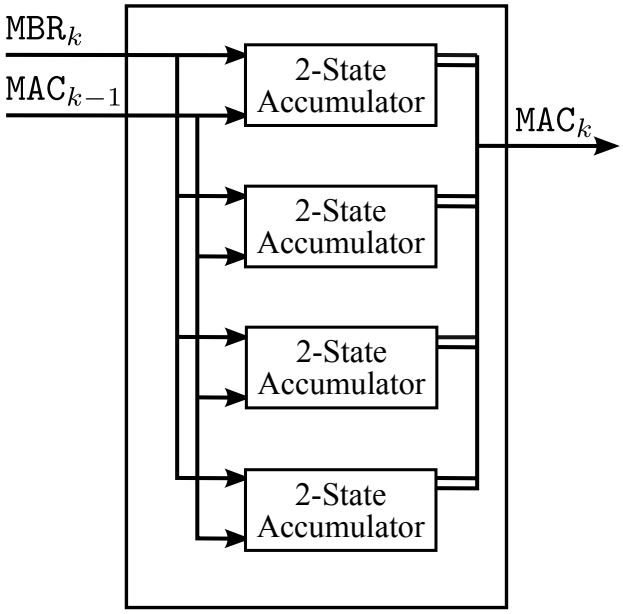


FIGURE 4.5: Schéma de calcul des métriques cumulées du module **AMP** dans le cas d'une convergence turbocode binaire et QC-LDPC

est établi par le module π_{BM} et dépend uniquement de la structure du treillis du code. De même les informations cumulées $\texttt{MAC}_k(s)$ sont entrelacées en fonction des états d'entrées et de sorties des structures de treillis génériques précédemment introduits. Le module π_s suit le cheminement d'ordonnancement des états en fonction des schémas de treillis génériques introduit sur la FIGURE 3.2b et sur la FIGURE 3.3b. Le changement de contexte de décodage s'effectue donc sur un cycle d'horloge, ce qui permet une forte flexibilité sur ce module. $\mathbf{b} = S/2$ éléments de calcul à deux états sont utilisés dans chaque cas pour une convergence entre un turbocode binaire et un code QC-LDPC et de $\mathbf{b} = S$ pour une convergence entre un turbocode double-binaire et un code QC-LDPC.

4.1.2.4 Module de calcul des décisions (DEC)

Les métriques cumulées $\texttt{MAC}_k(s)$ des sens *Aller* et *Retour* sont exploitées par le module de décision **DEC** pour estimer les vecteurs d'informations de décision et extrinsèques. Ce module reprend l'architecture présentée sur la FIGURE 2.15 et sur la FIGURE 2.17a du chapitre 2 pour la première et la dernière étape de calcul. L'étape intermédiaire aux ressources mutualisées est présentée sur la FIGURE 4.6. Le module renvoie en sortie du cœur SISO deux vecteurs d'informations correspondants l'un aux décisions prises sur les valeurs des informations systématiques de la section de treillis k représenté par le vecteur $\mathbf{\Lambda}_k$ et la valeur de l'information extrinsèque extraite du processus de décodage représentée par le vecteur d'information $\mathbf{Z}_k^{out}$. Ces vecteurs sont contraints aux types de codes pris en compte. La taille du vecteur $\mathbf{\Lambda}_k$ correspond au nombre d'éléments systématiques traités par section de treillis

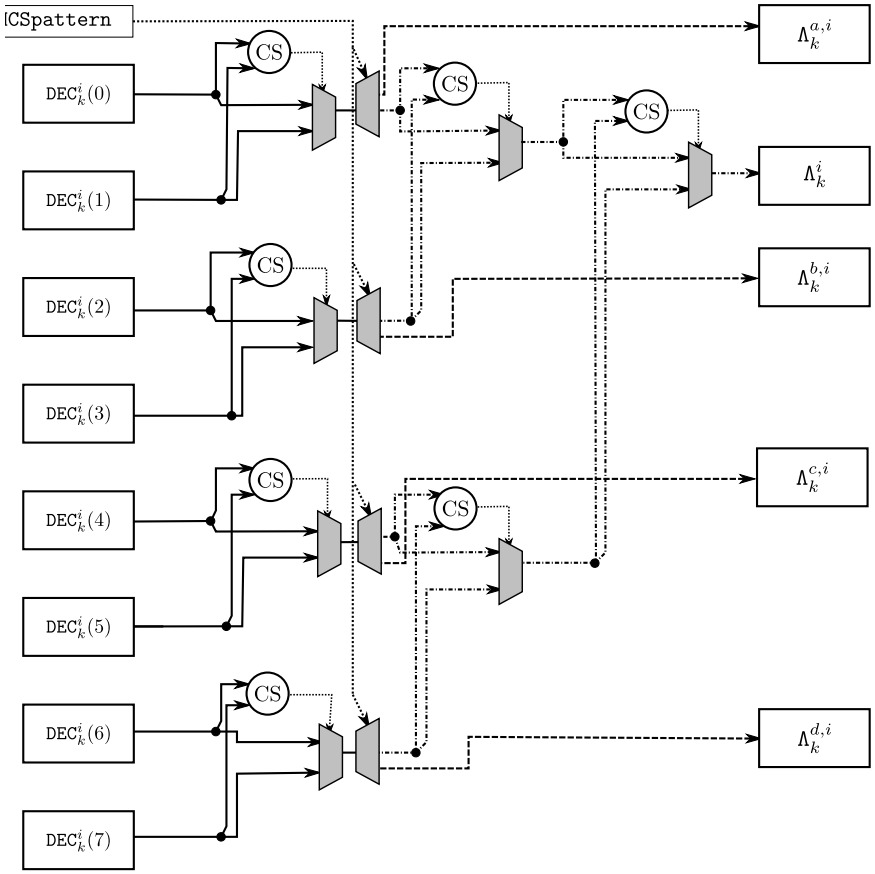


FIGURE 4.6: Schéma de calcul des décisions et informations extrinsèques du module **DEC** dans le cas d'une convergence turbocode binaire et QC-LDPC

(1 pour un turbocode binaire, 2 pour un turbocode double-binaire, et **b** pour un décodeur QC-LDPC). Pour le cas d'un décodage conjoint aux standards 3GPP LTE et IEEE 802.11n, ce vecteur se définit par la relation (4.3).

$$\mathbf{\Lambda}_k \triangleq \begin{pmatrix} \mathbf{\Lambda}_k^{\mathbf{0}} \\ \mathbf{\Lambda}_k^{\mathbf{1}} \\ \mathbf{\Lambda}_k^{\mathbf{2}} \\ \mathbf{\Lambda}_k^{\mathbf{3}} \end{pmatrix} = \begin{pmatrix} \mathrm{L}^a\left(s_k\right) \\ 0 \\ 0 \\ 0 \end{pmatrix}_{\text{LTE}} = \begin{pmatrix} \mathrm{L}^a\left(c_{\Pi(k,0)}\right) \\ \mathrm{L}^a\left(c_{\Pi(k,1)}\right) \\ \mathrm{L}^a\left(c_{\Pi(k,2)}\right) \\ \mathrm{L}^a\left(c_{\Pi(k,3)}\right) \end{pmatrix}_{\text{QC-LDPC}} \tag{4.3}$$

Le vecteur extrinsèque $\mathbf{Z}_k^{out}$ dépend du carré du nombre d'éléments systématiques m^2 par section pour le décodage de turbocodes et du nombre d'éléments systématiques **b** pour un décodage de codes QC-LDPC. Pour le cas d'un décodage conjoint aux standards 3GPP LTE et IEEE 802.11n, ce vecteur se définit par la relation (4.4).

$$\mathbf{Z}_k^{out} \triangleq \begin{pmatrix} \mathbf{Z}_k^{out,\mathbf{a}} \\ \mathbf{Z}_k^{out,\mathbf{b}} \\ \mathbf{Z}_k^{out,\mathbf{c}} \\ \mathbf{Z}_k^{out,\mathbf{d}} \end{pmatrix} = \begin{pmatrix} \mathrm{L}^e\left(s_k\right) \\ 0 \\ 0 \\ 0 \end{pmatrix}_{\text{LTE}} = \begin{pmatrix} \mathrm{L}^e\left(c_{\Pi(k,0)}\right) \\ \mathrm{L}^e\left(c_{\Pi(k,1)}\right) \\ \mathrm{L}^e\left(c_{\Pi(k,2)}\right) \\ \mathrm{L}^e\left(c_{\Pi(k,3)}\right) \end{pmatrix}_{\text{QC-LDPC}} \tag{4.4}$$

4.1.2.5 Paramétrage des mémoires internes du cœur SISO

Le cœur SISO nécessite deux types de bancs de mémoires **ACC RAM** (données $\mathtt{MAC}_k(s)$) et **SYST RAM** (données systématiques). Ces bancs fonctionnent à la manière de LIFO. Pour un ordonnancement du treillis *Retour* puis *Aller*, ces blocs de mémoires sont utilisés comme des mémoires tampons stockant l'information lors du parcours du treillis dans le sens *Retour* et en vidant la file dans le sens *Aller*. Le dimensionnement de ces mémoires est défini par la quantification des données stockées ainsi que par la taille maximale de la fenêtre de treillis w.

Une architecture ordonnancée en *Papillon* nécessite également d'utiliser ces types de RAM. La juxtaposition d'unités AMP *Aller* et *Retour* demande de doubler le nombre de RAM allouées dont la profondeur serait divisée par deux. Au final, la structure d'ordonnancement choisie ne modifie pas la quantité de stockage disponible, mais influence la synthèse des blocs mémoires alloués en fonction des différents outils de synthèse.

La taille des LIFO est conditionnée par la structure des codes en présence. Pour le décodage de codes QC-LDPC, le degré de parité maximal correspond à la taille maximale de la fenêtre. Pour les standards IEEE 802.11n et IEEE 802.11ac une profondeur de LIFO de 22 est suffisante. La Table 4.1 fournit les tailles de fenêtre maximales suivant certains standards et donc la profondeur de RAM nécessaire à l'architecture de décodage associée.

L'entrelacement du turbocode a été étudié dans le chapitre 3. Le fenêtrage doit être analysé afin de dimensionner la profondeur de la RAM pour ce cas d'usage. Le chapitre 3 (section 3.3.2) montre que pour des codes 3GPP LTE une taille de fenêtre

Standard	Facteur d'expansion z	Taille de fenêtre maximale
802.11n [14]	$27, 54, 81$	22
802.11ac [15]	$27, 54, 81$	22
802.11ad [16]	42	16
802.16m [17]	$24, ..., 96$	20
802.15.3c [18]	21	32
G.Hn [19]	$14, 48, 60, 80, 216, 170, 360$	21

TABLE 4.1: Taille de fenêtre maximale en fonction des codes QC-LDPC standardisés

de 32 est intéressante du point de vue des performances de décodage et respecte également les contraintes d'accès aux mémoires en fonction de son entrelaceur.

4.1.2.6 Paramétrage du nombre de blocs d'états suivant le turbocode

Pour un décodage conjoint avec une mutualisation des ressources de calcul, la forme du *treillis d'itération* lié à la matrice de parité du code QC-LDPC est contrainte à la caractéristique du code convolutif constituant du turbocode associé. Si S représente le nombre d'états caractérisant le code convolutif associé au turbocode, alors le nombre maximal d'équations de parité du code QC-LDPC traitées en parallèle par le cœur SISO est de $\mathbf{b} = S/2$. Dans ce cas, il est possible de dénombrer les composantes du cœur SISO à l'aide du facteur de contrainte $\mathbf{b}$. Les LIFO doivent stocker $2.\mathbf{b}$ métriques cumulées et $\mathbf{b}$ messages systématiques par section de treillis dans le cadre d'un décodage QC-LDPC. Ainsi, le choix du facteur de contrainte doit être effectué en prenant en compte les caractéristiques des turbocodes en présence. La TABLE 4.2 regroupe les caractéristiques de plusieurs turbocodes standardisés. Un facteur de contrainte y est également indiqué en prenant en compte les critères expliqués précédemment.

4.1.2.7 Reconfiguration du cœur SISO et débits de décodage

Le cœur de processeur SISO est une architecture complètement séquencée. Elle est synchronisée suivant une fréquence d'horloge f_{clk} dont la période est notée T_{clk}. Pour cette architecture, et avec un ordonnancement en *Papillon*, les informations relatives à une section de treillis sont calculées par le cœur à chaque cycle d'horloge en régime permanent. On réintroduit le nombre d'informations utiles m défini dans le chapitre 1. Pour le décodage d'un treillis dont la taille de la fenêtre est infinie, le débit de traitement d'un cœur SISO noté D^{SISO} vérifie alors asymptotiquement la relation (4.5) en régime permanent.

$$D^{SISO} = m \times f_{clk} \tag{4.5}$$

Standard	Type de turbocode convolutif	Nombre d'états S	Facteur de contrainte b
CCSDS [20]	binaire	16	8
UMTS [21]	binaire	8	4
LTE/LTE-A [22]	binaire	8	4
802.16m [17]	double-binaire	8	8
DVB-RCS [24]	double-binaire	8	8
DVB-RCT [25]	double-binaire	8	8
DVB-SH [26]	binaire	8	4
DVB-RCS2 [27]	double-binaire	16	16
HPAV [28]	double-binaire	8	8

TABLE 4.2: Paramétrage du facteur de contrainte **b** du code QC-LDPC en fonction du turbocode

Le traitement d'un treillis de taille w réduit le débit de décodage du cœur de processeur SISO. Le temps de parcours du treillis suivant un ordre de calcul en *Papillon* dépend tout d'abord de la parité de cette fenêtre. Pour répondre aux contraintes de synchronisation du décodage, les processus *Aller* et *Retour* doivent être décaller d'une section de décodage et donc d'un cycle d'horloge. Ainsi, le temps de parcour du treillis est de $2.\lceil \frac{w}{2} \rceil$ pour toutes tailles de fenêtre. Une architecture séquencée permet de libérer l'utilisation des premiers modules avant l'obtention des informations sur le treillis en cours de calcul. De ce fait, ces modules peuvent être exploités pour initialiser le calcul d'un autre treillis, tout en gardant une latence de δ_T cycles d'horloge pour modifier les conditions au sein du cœur SISO. Dans ce cas, le débit du cœur SISO vérifie (4.6).

$$D^{SISO} = \frac{m \times w}{2.\lceil \frac{w}{2} \rceil + \delta_T} \times f_{clk} \tag{4.6}$$

Le délai de latence de reconfiguration δ_T permet de contrôler le décodage par les modules externes au cœur de décodage, de modifier les métriques cumulées initiales et de modifier l'architecture en fonction de la structure du treillis. Pour une architecture séquencée, un délai d'un cycle d'horloge permet de modifier ces conditions, à condition que le délai δ_{pip} soit inchangé. En effet, la flexibilité du cœur SISO étant déterminé par des changements de route avec quelques multiplexeurs, il suffit de retarder le jeton de changement de contexte pour éviter de perturber le décodage des éléments en cours. Cependant, si le délai δ_{pip} est modifié, le changement de contexte peut entraîner des conflits d'accès aux ressources de calcul, ce qui compromet le calcul des éléments en bordure de ces treillis. Pour éviter ces phénomènes, il est préférable de fixer δ_T en fonction de δ_{pip} avant un changement de structure du décodeur SISO. Le chronogramme de la FIGURE 4.2b prend en compte cette contrainte.

Lorsque le cœur SISO décode le code convolutif constituant le turbocode 3GPP

LTE, 9 périodes d'horloge sont nécessaires pour vider les tuyaux du décodeur sans incidence, contre 8 périodes pour tout type de codes convolutifs constituant les codes QC-LDPC. Cette différence de latence est dûe au choix de synchronisation élaboré sur le module **DEC** dont le formalisme a été représenté sur la FIGURE 4.6.

4.1.3 Application au cas d'usage 3GPP LTE et IEEE 802.11n

Une architecture de décodage conjoint a été testée pour un cas de convergence de standards 3GPP LTE et IEEE 802.11n. L'architecture suit un ordonnancement du treillis en *Papillon*. On se propose de vérifier les impacts de l'usage d'une telle architecture SISO sur le décodage de treillis de longueur w pour un tronçon de treillis 3GPP LTE et un tronçon IEEE 802.11n. La taille du tronçon est définie en fonction de la taille de la fenêtre pour le cas du turbocode et par le degré de parité pour le cas QC-LDPC.

Le schéma général du cœur SISO suit le schéma de la FIGURE 4.3b. La quantification des signaux internes est directement liée à la quantification des données d'entrée et de sortie du cœur SISO. On reprend dans cette partie les notations données dans le chapitre 2. Les données intrinsèques fournies par le vecteur $\mathbf{X}_k$ suivent une quantification signée en virgule fixe de la forme $Q4.6$. Ces signaux sont donc codés sur 10 bits avec une dynamique de 6 bits pour la partie fractionnaire. Ce choix est effectué en fonction des critères de quantification établis dans le chapitre 2. Une quantification moindre est sélectionnée pour les données du vecteur d'informations extrinsèques $\mathbf{Z}_k^{in}$. Ces informations sont quantifiées sur 8 bits avec une dynamique de 3 bits pour la partie décimale et 5 bits pour la partie fractionnaire. La dimension de ces vecteurs a été étudiée afin de réduire cette dynamique sans dégradation du BER.

Du fait de la profondeur des informations incidentes, les métriques de branches $\texttt{MBR}_k()$ en sortie du module **BMP** sont échelonnées sur 10 bits suivant la dynamique attribuée aux informations intrinsèques. La dynamique des métriques cumulées $\texttt{MAC}_k(s)$ a été étudiée pour une quantification minimale. Cette valeur est initialisée avec les valeurs 32.0 pour le décodage QC-LDPC et 16.0 pour le décodage 3GPP LTE, puis codifiée selon la même dynamique que l'information intrinsèque. Afin d'éviter les phénomènes de débordement des métriques, un seuil est fixé à la valeur 8.0. De ce fait, toutes les valeurs $\texttt{MAC}_k(s)$ sont réduites si celles-ci sont toutes supérieures à 8.0 dans le cas du fonctionnement turbocode, et suivant les états $2.s$ et $2.s + 1$ pour le cas QC-LDPC.

Les informations de décision en sortie conservent la quantification des informations intrinsèques ($Q4.6$). Le facteur d'échelonnage de l'information de décision f_Λ choisi est 0.75. Afin de limiter les dégradations dues à la multiplication par le facteur inverse ($f_Z = 1.33$), la dynamique des signaux est modifiée localement suivant la quantification $Q4.10$. Pour le cas d'usage 3GPP LTE, par contre, seule l'information extrinsèque est échelonnée suivant les paramètres définis précédemment.

L'architecture du cœur SISO a été synthétisée pour une carte FPGA Virtex6 (6VLX75T) afin d'avoir la preuve de concept du cœur SISO. La TABLE 4.3 résume

ces résultats de synthèse pour les quantifications décrites dans ce paragraphe. L'architecture du cœur SISO est comparée à titre informatif avec une architecture SISO pour un décodeur SISO compatible uniquement avec le code 3GPP LTE et un décodeur SISO compatible uniquement avec le code QC-LDPC avec un facteur de parallélisme **b** de 4. Ce tableau montre ainsi le surplus de ressources matérielles nécessaire à la double compatibilité de standard par rapport à une architecture simple standard suivant le même algorithme de décodage. Le décodage QC-LDPC augmente le chemin critique de l'architecture à cause du facteur d'échelonnage appliqué en sortie du décodeur. La quantité de RAM nécessaire est optimisée et le surcoût de ressources matérielles est moindre que deux architectures dédiées côte à côte.

	cœur SISO conjoint	**cœur SISO LTE**	**cœur SISO 802.11n**
Architecture avec ordonnancement du treillis en *Papillon*			
Utilisation de slices logiques			
Slice Registers	2245 (2 %)	1788 (1 %)	2044 (2 %)
Slice LUTs	4330 (7 %)	3368 (5 %)	3294 (6 %)
Distribution des slices logiques			
LUT Flip Flop pairs	4444	3419	3404
Block RAM/FIFO			
Block RAM	6 (3 %)	6 (3 %)	6 (3 %)
Fréquence			
Fréquence	232 MHz	266 MHz	239 MHz

TABLE 4.3: Synthèse du cœur SISO compatible avec le standard 3GPP LTE seul, IEEE 802.11n seul et pour les deux standards, pour des architectures avec ordonnancement du treillis en *Papillon*, sur une carte Virtex 6 (6VLX75T)

Cette architecture a été testée pour une convergence de standard 3GPP LTE et IEEE 802.11n. De plus, ce cœur SISO est compatible pour le décodage de toute matrice QC-LDPC à matrice identité circulante dont le degré de parité maximal ne dépasse pas 32. À partir de ces données, le débit utile du cœur SISO atteint 800 Mbps pour le décodage de treillis IEEE 802.11n et 220 Mbps pour le décodage de codes 3GPP LTE.

4.2 Architectures de décodage itératif de turbocodes et de codes QC-LDPC

Une architecture de cœur SISO a été présentée pour répondre à plusieurs scénarios de convergence de standard dans la section 4.1. Cette architecture est reprise pour la conception d'une architecture de décodage générique et flexible. Dans une première partie, une architecture à un cœur SISO propose la gestion de l'entrelace-

ment particulier des codes QC-LDPC et décrit la problématique de partage d'accès aux mémoires. Une seconde architecture multi-cœurs est proposée afin de bénéficier du parallélisme offert par chaque code et d'améliorer les débits de décodage. L'impact de cette architecture sur la gestion des mémoires est également abordé.

4.2.1 Architecture de décodage à un cœur

4.2.1.1 Contraintes de l'architecture

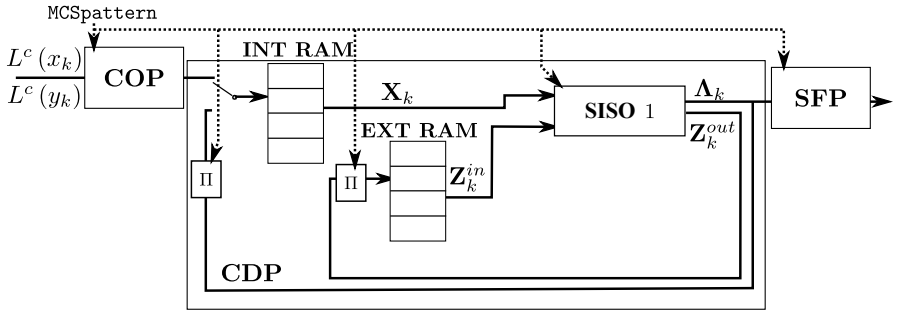


FIGURE 4.7: Architecture de décodage à un cœur SISO

Une architecture de décodage à un cœur SISO permet d'obtenir un décodage efficace avec des débits minimaux. Cette architecture doit résoudre les problématiques liées aux disparités entre les turbocodes et les matrices QC-LDPC. L'architecture de décodage à un cœur SISO est générique. Le cœur est supposé être compatible avec les codes envisagés. De ce fait, il reste à définir les types de mémoires associées au décodeur, à générer un entrelaceur compatible avec les turbocodes et les codes QC-LDPC et à vérifier les contraintes de conflits d'accès pour les codes QC-LDPC. Cette dernière discussion permet de classifier les codes QC-LDPC suivant leur facteur d'expansion z. Dans un souci d'efficacité, le décodeur est scindé en trois parties indépendantes correspondant aux phases successives du traitement. Un premier module **COP** (*Codeword Ordering Processor*) a pour but de gérer l'ordonnancement des informations en sortie du démodulateur. Ce module doit donc avoir un accès dédié et indépendant aux RAM d'information du canal. Un second module **CDP** (*Codeword Decoding Processor*) applique la solution de décodage aux messages reçus. Ce module doit fonctionner de façon indépendante et être flexible pour un décodage à la volée. Le nombre d'itérations et la forme du treillis sont définis en fonction du code traité. Un troisième module **SFP** (*Systematic Flush Processor*) organise la sortie de la décision dure en fin de décodage. La FIGURE 4.7 illustre l'architecture générique du décodeur de canal avec un seul cœur SISO.

Dans la suite de cette partie, l'architecture conjointe fait appel aux codes associés aux standards 3GPP LTE concernant l'application turbocode et au standard IEEE 802.11n concernant l'application QC-LDPC. Pour ce cas de convergence, le facteur de contrainte **b** est fixé à 4.

4.2.1.2 Gestion des mots de code incidents et de la sortie de décision

Une architecture au rendement de décodage efficace doit pouvoir gérer l'arrivée d'un mot de code en parallèle de l'étape de décodage. Pour ce faire, on propose de dédoubler les bancs de mémoire. Une machine d'état permet de mettre à disposition l'un des bancs mémoires pour effectuer le décodage d'un mot de code, l'autre étant disponible en cas de réception d'un mot différent. Cette machine d'état doit donc également gérer la mémorisation des paramètres de décodage permettant de connaître le standard et le schéma de codage associé. L'architecture des trois modules de réception d'un message **COP**, de décodage **CDP** et d'extraction des données systématiques **SFP** du décodeur permet d'optimiser le débit du décodage.

La communication et les temps d'occupation des différents modules du décodage sont illustrés sur la FIGURE 4.8. Le module **CDP** est entièrement dédié au décodage d'un mot de code. Dans la suite de cette section, on suppose que ce module fonctionne à flux tendu. De ce fait, le débit de décodage est défini en fonction du débit de traitement de ce module. La latence nécessaire aux opérations effectuées par les autres modules **COP** et **SFP** est négligée.

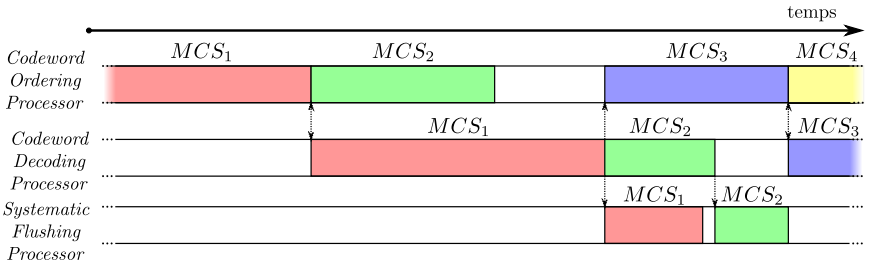


FIGURE 4.8: Ordre des opérations du décodeur

La mission principale du module **COP** est de désentrelacer le message en provenance du démodulateur en fonction du treillis équivalent aux codes étudiés. La réception du mot de code d'un MCS turbocode doit distinguer les informations systématiques et les informations de parité par rapport aux deux dimensions de décodage (naturel et entrelacé). La réception d'un mot de code d'un MCS QC-LDPC doit appliquer la transformation en treillis de la matrice de parité associée en fonction des procédés établis dans le chapitre 3 (section 3.2). Pour chaque usage, on suppose que le démodulateur fournit un train d'informations de Log-Rapports de Vraisemblance suivant l'ordre croissant $\mathrm{L}^{c}(c_n)$. Les messages incidents des mots de codes QC-LDPC donnent les informations du canal sur les variables c_n dans le sens croissant. Les MCS turbocodes suivent l'ordonnancement défini par les standards.

4.2.1.3 Gestion des bancs de mémoire

L'organisation du banc de mémoire **INT RAM** suppose de connaître l'organisation des treillis équivalents. L'adresse de l'information stockée est établie en fonction de son action dans le treillis équivalent proposé dans le chapitre 3.

Pour une architecture de décodage compatible avec les standards 3GPP LTE et IEEE 802.11n, le cœur SISO doit accéder à un vecteur d'information $\mathbf{X}_k$ de 4 valeurs. De ce fait, la mémoire **INT RAM** est scindée en $\mathbf{b} = 4$ bancs de mémoires repérés par les valeurs $\mathtt{MEM_a}$, $\mathtt{MEM_b}$, $\mathtt{MEM_c}$ et $\mathtt{MEM_d}$. Pour les mêmes raisons, le banc de mémoire extrinsèque **EXT RAM** bénéficie de la même découpe. Dans les deux cas, la profondeur de RAM est définie en fonction des longueurs de treillis équivalent pour une itération de décodage T_{It}.

Pour le cas d'usage 3GPP LTE, seuls deux bancs de données intrinsèques **INT RAM** sont utiles si on considère que les décodages des treillis entrelacés et naturels sont séquentiels. La profondeur de ces bancs est alors de $2 \times T_l$. Afin de mutualiser les ressources avec l'usage IEEE 802.11n, il est préférable de séparer ces informations suivant les ordres entrelacé et naturel. Les Log-Rapports de Vraisemblance du démodulateur relatif au décodeur naturel sont stockés dans les bancs $\mathtt{MEM_a}$ $(\mathrm{L}^c(s_k^n))$ et $\mathtt{MEM_b}$ $(\mathrm{L}^c(p_k^n))$ à l'adresse naturelle. Les bancs $\mathtt{MEM_c}$ conservent alors l'information systématique $(\mathrm{L}^c(s_k^i))$ enregistrée dans l'ordre désentrelacé à l'adresse $\Pi^{-1}(k)$, et le banc $\mathtt{MEM_d}$ l'information de parité associée $\mathrm{L}^c(p_k^i)$. L'enregistrement des données intrinsèques et extrinsèques du treillis suivent l'ordre κ défini sur le *treillis d'itération*. L'usage des bancs de mémoires $\mathtt{MEM_c}$ et $\mathtt{MEM_d}$ permet dans ce cas de réduire la profondeur de RAM pour améliorer la mutualisation de ressources entre les usages.

De même, seul un banc de mémoire pour les informations extrinsèques **EXT RAM** sont utiles. De ce fait, les données extrinsèques utiles pour le décodage dans l'ordre naturel sont stockées dans le banc de mémoire $\mathtt{MEM_a}$ et celles pour le décodage entrelacé dans le banc de mémoire $\mathtt{MEM_c}$.

4.2.1.4 Gestion de l'adressage des mémoires

Pour le cas d'usage 3GPP LTE, le bloc d'entrelacement des adresses calcule l'adresse d'enregistrement de l'information extrinsèque dans l'ordre entrelacé (pour les données en provenance du décodeur naturel) et désentrelacé (pour les données en provenance du décodeur entrelacé). Ces adresses sont calculées à la volée. En effet, l'adresse d'entrelacement QPP associé au standard 3GPP LTE s'exprime suivant la relation (4.7).

$$\begin{aligned} \Pi(k+1) &= \left(f_1.k + f_2.k^2 + \underbrace{2.f_2}.k + \underbrace{f_1 + f_2} \right)_{[K]} \\ \Pi(k+1) &= \left(\Pi(k)\right)_{[K]} + \left(P_b.k\right)_{[K]} + P_a \end{aligned} \tag{4.7}$$

Seules les métriques P_a et P_b sont utiles au calcul de l'adresse d'entrelacement. Cette adresse est donc calculé à la volée.

Le calcul de l'adresse d'un mot de code QC-LDPC est plus complexe. En effet, pour chaque valeur de l'information incidente $\mathrm{L}^c(c_n)$ il existe un couple de valeur (κ, b) dont la caractéristique a été établie au chapitre 3. Dans ce cas, b correspond à l'indice du banc de mémoire (enregistrement de l'information dans le banc $\mathtt{MEM_a}$ si $b = 0$, $\mathtt{MEM_b}$ si $b = 1$, ...). Les informations du démodulateur sont fournies dans l'ordre croissant, le calcul du couple (κ, b) est alors simplifié.

On s'intéresse au calcul de la relation entre les positions des variables n et les indices de treillis (κ, b) d'un ensemble d'équations du groupe de variables de l'ensemble$[\![\mathbf{n}.z; (\mathbf{n}+1).z[\![$ pour les équations du groupe $\mathcal{C}^{\mathbf{m_1}}$. On pose ζ tel que $\zeta \in [\![\mathbf{m_1}.z; (\mathbf{m_1}+1).z[\![$. On suppose que la sous-matrice carrée de position $(\mathbf{m_1}, \mathbf{n})$ n'est pas nulle. De ce fait, il existe un indice de circulation pour cette matrice que l'on note $j_{\mathbf{m_1},\mathbf{n}}$. La relation entre l'indice de variable n et les indices de treillis (κ, b) est définie par itération sur ζ.

On pose (κ_0, b_0) comme le couple correspondant à l'indice de treillis pour la variable $\mathbf{n}.z$. D'après les relations (3.14) et (3.25) établies au chapitre 3 :

$$\begin{array}{rcl} \kappa_0 & = & T_l(m) + \pi_0^{-1}(\mathbf{n}.z) \\ b_0 & = & (z - j_{\mathbf{m_1},\mathbf{n}})_{[z]} \end{array}$$

Le système (4.8) permet alors de retrouver la correspondance avec la position du treillis par récurrence.

$$\Pi(\mathbf{n}.z+\zeta) = \begin{cases} \left(\kappa_{\zeta-1} - T_l^{\mathbf{m_1}}, 0\right) & \text{si } \zeta = j_{\mathbf{m_1},\mathbf{n}} \\ \left(\kappa_{\zeta-1} + \left\lfloor \frac{b_{\zeta-1}}{\mathbf{b}} \right\rfloor . d_c^{\mathbf{m_1}}, (b_{\zeta-1}+1)_{[\mathbf{b}]}\right) & \text{sinon} \end{cases} \tag{4.8}$$

Le mot de code incident est donc mémorisé en fonction des N/z groupes d'informations successifs.

4.2.1.5 Génération des adresses d'entrelacement

L'entrelacement concerne deux stratégies distinctes pour le décodage d'un turbocode et le décodage d'un code QC-LDPC.

Le décodage d'un turbocode nécessite de faire transiter des informations extrinsèques entre les décodeurs. Ces informations estiment le gain apporté par le décodage dans l'ordre naturel ou entrelacé à une itération donnée. Ce gain est ensuite injecté dans le décodeur concurrent afin d'améliorer sa prédiction en fonction de l'information sur la variable (code convolutif constituant binaire) ou sur le couple (code convolutif constituant double-binaire). Le cœur de processeur doit connaître la confiance intrinsèque provenant du démodulateur et la dernière information extrinsèque relative à une section de treillis. De ce fait, les données extrinsèques contenues dans le vecteurs $\mathbf{Z}_k^{in}$ doivent être enregistrées à l'adresse entrelacée $\Pi(k)$ en sortie du cœur de décodage naturel et à l'adresse dé-entrelacée $\Pi^{-1}(k)$ en sortie du cœur de décodage entrelacé.

Le vecteur d'informations intrinsèques $\mathbf{X}_k$ utilisé par le cœur de décodage SISO reprend la dernière mise à jour des variables pour l'usage QC-LDPC. De ce fait, il contient à la fois l'information fournie par le canal et l'information extrinsèque provenant de la dernière mise à jour. La multiplicité des dimensions du code QC-LDPC requiert cependant de retrancher à cette donnée l'information extrinsèque provenant du décodage de la même équation à l'itération précédente. De ce fait, pour le décodage d'un code QC-LDPC, le vecteur $\mathbf{Z}_k^{out}$ agglomère les données extrinsèques

extraites de l'itération précédente. Le vecteur est stocké dans une RAM à la manière d'une FIFO et ne nécessite pas de calcul d'entrelacement particulier.

En revanche, la dernière décision de décodage est réinjectée dans le processus de décodage, et dans ce cas d'usage, une fonction d'entrelacement est nécessaire pour faire correspondre les composantes du vecteur de décision en sortie du cœur SISO $\mathbf{\Lambda}_{\kappa_1}^{\mathbf{m_1}}$ à la prochaine réutilisation de la variable c_n associée à cette décision et sa prochaine implication dans le treillis équivalent. L'entrelacement QC-LDPC doit donc effectuer les opérations de transfert décrites dans le chapitre 3.

Le calcul de l'entrelacement entre deux sous-groupes d'équations $\mathcal{C}^{\mathbf{m_1}}$ et $\mathcal{C}^{\mathbf{m_2}}$ suit la logique du calcul de l'entrelacement initial. On pose $\zeta_2 = (\zeta + j_{\mathbf{m_1,n}})_{[z]}$. La relation (4.8) s'adapte dans ces conditions avec la relation (4.9).

$$\Pi(n.z+\zeta_2) = \begin{cases} (T_l^{\mathbf{m_2}}, 0) & \text{si} \quad \zeta_2 = j_{\mathbf{m_2,n}} \\ (\kappa_{\zeta_2-1} + \left\lfloor \frac{b_{\zeta_2-1}}{\mathbf{b}} \right\rfloor . d_c^{\mathbf{m_2}}, (b_{\zeta_2-1}+1)_{[\mathbf{b}]}) & \text{sinon} \end{cases} \tag{4.9}$$

Dans ce cas,

$$b_{\zeta=0}^{\mathbf{m_2}} \quad = \quad (z + j_{\mathbf{m_1,n}})_{[z]} - (j_{\mathbf{m_2,n}})_{[z]} \tag{4.10}$$

4.2.1.6 Conflits d'accès dûs aux différents usages

Dans le paragraphe précédent, il a été établi que l'entrelacement de l'information extrinsèque dépend d'une mise en mémoire de type FIFO pour le décodage QC-LDPC. Le décodage de codes QC-LDPC n'a donc pas à faire face à des contraintes de conflits d'accès pour les mémoires **EXT RAM**. Par contre, ce banc mémoire est contraint à la fonction d'entrelacement $\Pi(k)$ et la fonction d'entrelacement inverse $\Pi^{-1}(k)$ du turbocode. Les risques de conflits d'accès sont donc causés par cet entrelaceur. Pour un code 3GPP LTE, la fonction d'entrelacement est sans conflit d'accès tout comme sa fonction inverse. De ce fait, la RAM de données extrinsèques ne souffre pas de conflit d'accès pour une architecture à un cœur.

Le banc de mémoires d'informations intrinsèques **INT RAM** est mis à jour uniquement par le module d'ordonnancement du mot de code **COP** pour un cas d'usage turbocode. Ce cas n'entraîne pas de conflit d'accès. Le décodage QC-LDPC effectue des mises à jour permanentes sur ce banc de mémoire.

La fonction d'entrelacement liée au code et définie dans le chapitre 3 nécessite le calcul d'une adresse couple (κ, b) pour chaque composante du vecteur de décision $\mathbf{\Lambda}_k$ sortie du cœur de processeur, dont les relations sont données par la fonction (4.9). Pour résoudre les risques de conflits d'accès, il suffit de scinder le banc de mémoire en $\mathbf{b}$ RAM indépendantes. La sélection de la RAM est alors donnée par l'information b de la fonction d'entrelacement. Cette information dépend des indices de permutation des matrices identités circulaires $j_{\mathbf{m,n}}$. L'obtention du facteur b suit une loi incrémentale modulo $\mathbf{b}$ dans la majorité des cas. Une remise à zéro est cependant nécessaire à l'indice ζ_x vérifiant (4.11).

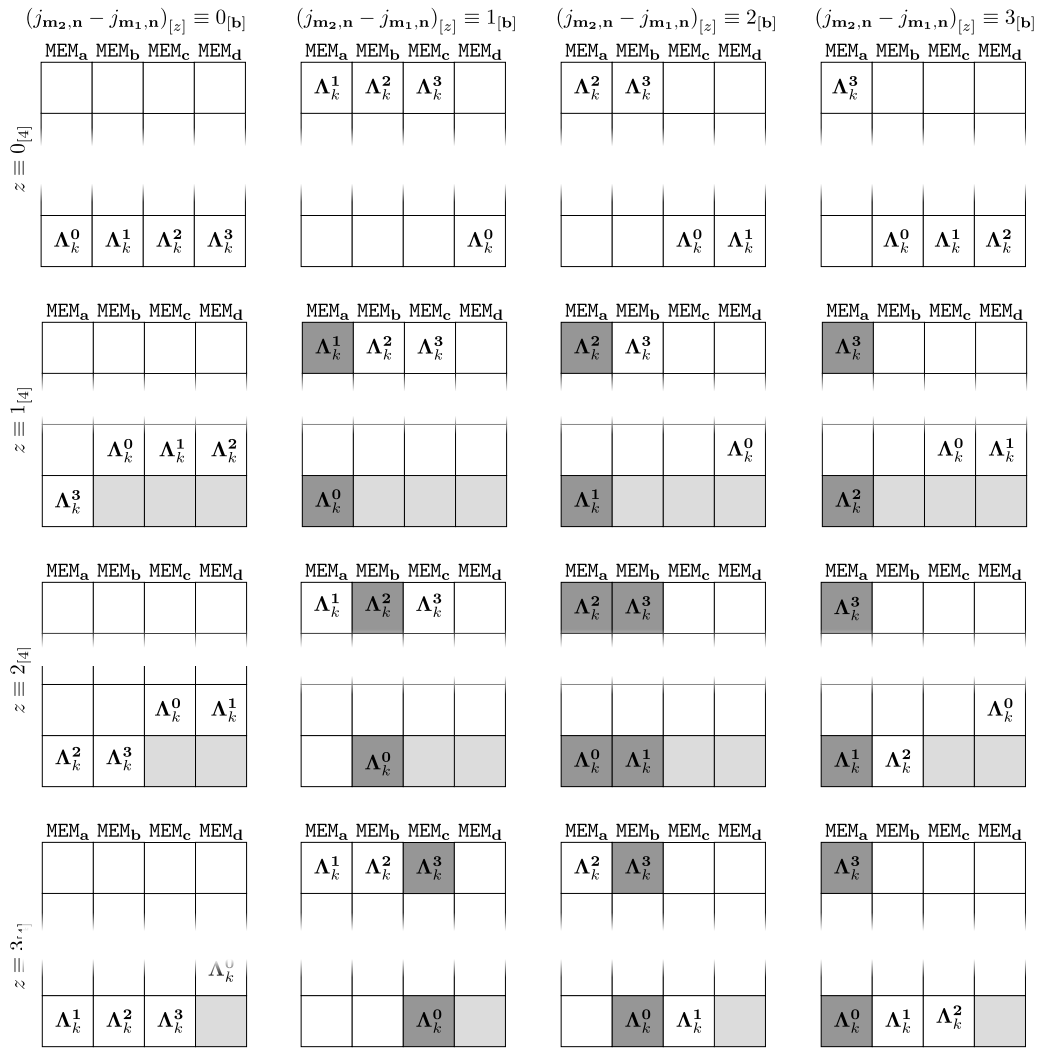


FIGURE 4.9: Illustration de conflits d'accès pour l'enregistrement des mises à jour de la section k en fonction des contraintes de la matrice QC-LDPC

$$\zeta_2 = j_{\mathbf{m_2},\mathbf{n}} \quad \Leftrightarrow \quad \zeta_x \equiv (j_{\mathbf{m_2},\mathbf{n}} - j_{\mathbf{m_1},\mathbf{n}})_{[z]} \tag{4.11}$$

Une condition suffisante pour qu'il y ait continuité de l'incrément est obtenue dans le cas où z est divisible par $\mathbf{b}$. En effet, dans ce cas, (4.12) est vérifiée.

$$\begin{aligned} b_{\zeta_x - 1} &= (b_0 + \zeta_x - 1)_{[\mathbf{b}]} \\ &= \left(b_0 + (j_{\mathbf{m_2},\mathbf{n}} - j_{\mathbf{m_1},\mathbf{n}})_{[z]} - 1\right)_{[\mathbf{b}]} \end{aligned} \tag{4.12}$$

Or d'après (4.10), $b_{\zeta=0}^{\mathbf{m_2}} = (z + j_{\mathbf{m_1},\mathbf{n}} - j_{\mathbf{m_2},\mathbf{n}})_{[z]}$ donc

$$\exists x \in \mathbb{N},\ b_0 = (z.x + j_{\mathbf{m_1},\mathbf{n}} - j_{\mathbf{m_2},\mathbf{n}})_{[z]}$$

En remplaçant cette expression dans (4.12), on obtient (4.13)

$$b_{\zeta_x - 1} \quad = \quad (z.x - 1)_{[\mathbf{b}]} \tag{4.13}$$

Dans ce cas, $b_{\zeta_x - 1} = \mathbf{b} - 1$ si et seulement si (4.14)

$$(z.x)_{[\mathbf{b}]} \quad \equiv \quad (0)_{[\mathbf{b}]} \tag{4.14}$$

Cette propriété est vérifiée dans tout les cas si $z = (0)_{[\mathbf{b}]}$. Il dépend de la valeur du facteur x sinon.

Si la condition $b_{\zeta_x - 1} = \mathbf{b} - 1$ est vérifiée, à chaque section de treillis correspond un et un seul indice de banc b. Il n'y a donc pas de conflit d'accès. Lorsque $\mathbf{b}$ ne divise pas z, deux composantes de $\mathbf{\Lambda}_k$ peuvent être enregistrées dans la même RAM et entraîner des conflits d'accès. La FIGURE 4.9 représente les cas de conflits d'accès à un banc de mémoire pour $\mathbf{b} = 4$. Sur un même temps d'horloge, toutes les composantes du vecteur $\mathbf{\Lambda}_k$ doivent être enregistrées sur des bancs de mémoires d'indice b à des adresses κ définies par le calcul récursif résumé par l'équation (4.8). Certaines configurations n'entraînent pas de conflit d'accès, en particulier si $j_{\mathbf{m_2},\mathbf{n}} - j_{\mathbf{m_1},\mathbf{n}} \equiv (0)_{[\mathbf{b}]}$. Dans les autres cas, un même banc mémoire est accédé au plus deux fois pour une section de treillis donnée. Sur la FIGURE 4.9, les cas de conflits d'accès sont représentés par des emplacements grisés. La configuration du conflit dépend dans ce cas du facteur d'expansion z et des indices de circulation de la matrice de parité. Cette propriété est généralisée pour tout facteur de contrainte $\mathbf{b}$.

Deux stratégies permettent de résoudre ce conflit d'accès. Dans un premier temps, on considère que l'ordonnancement du treillis du cœur SISO est sous la forme *Retour* puis *Aller* (ou *Aller* puis *Retour*). Dans ce cas, les accès aux bancs en lecture et en écriture ne sont pas effectués sur le même cycle d'horloge et une RAM simple port suffit à gérer l'accès aux données dans un cas sans conflit d'accès. Le passage de ces bancs de mémoire en double-ports règle ponctuellement les conflits d'accès, puisqu'une même RAM sera accédée au plus deux fois au même cycle d'horloge pendant la phase de décodage.

Dans le cas d'un ordonnancement de treillis en *Papillon*, le banc de mémoire peut nécessiter entre zéro et quatre écritures sur le même cycle d'horloge, ce qui est source de conflits d'accès. En décalant la mise à jour de ces données, le conflit d'accès est résolu. La FIGURE 4.10 représente le décodage parallèle de deux *treillis de contrainte de parité* suivant un ordonnancement de treillis en *Papillon*. Nous observons ici une structure de décodage pour $\mathbf{b} = 2$. Sur ce cas de figure, les composantes $\mathbf{\Lambda}_1^{\mathbf{a}}$ et $\mathbf{\Lambda}_1^{\mathbf{b}}$ sont enregistrées sur le même banc $\mathtt{MEM}_{\mathbf{a}}$, ce qui entraîne un conflit d'écriture sur ce banc mémoire. Pour contrer ce phénomène, la composante $\mathbf{\Lambda}_1^{\mathbf{b}}$ est temporairement stockée en mémoire FIFO pour être positionnée sur le banc mémoire en fin de décodage des deux *treillis de contrainte de parité*. Ce cas de conflit est positionné sur la figure par un cercle. Pour réaliser cette stratégie, une FIFO de profondeur maximale w par banc mémoire doit être ajoutée afin de contenir tout cas de conflits d'accès. Cette technique augmente donc les ressources matérielles et entraîne une latence supplémentaire pour vider la mémoire FIFO en fin de décodage.

Une autre stratégie non exploitée ici consiste à ne pas effectuer la mise à jour de la variable en question. Cependant, cette stratégie entraîne une perte de performance due à un arrêt de transmission des mises à jour pour les variables incriminées sur chaque itération.

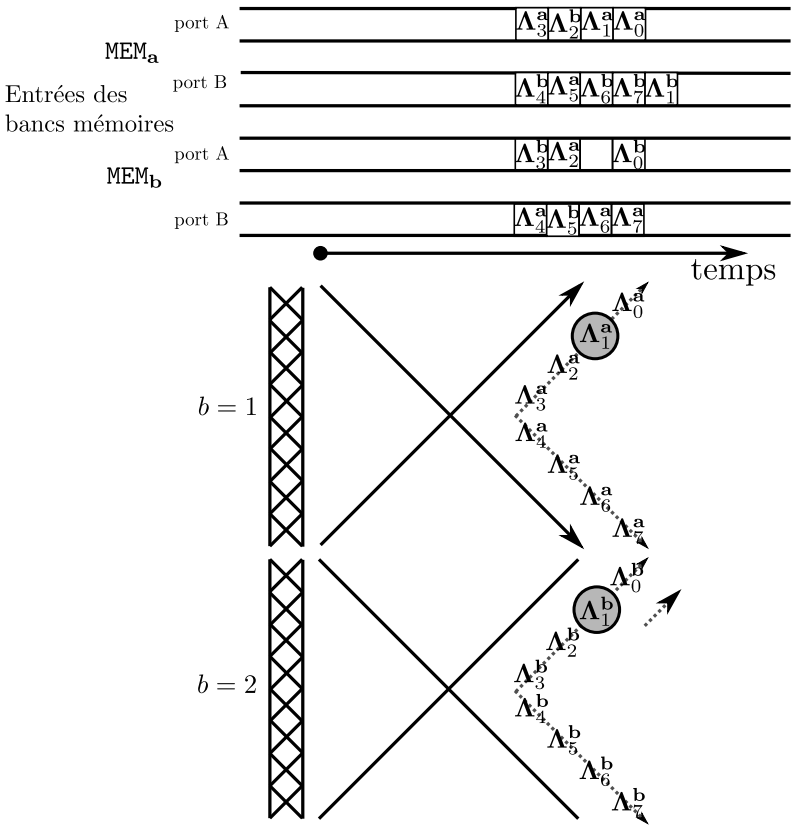


FIGURE 4.10: Représentation du conflit d'accès en écriture des bancs de mémoires intrinsèques dans le cas d'un cœur de processeur SISO en ordonnancement en *Papillon*

4.2.1.7 Gestion du transfert des métriques entre itérations

Entre chaque traitement de tronçons de treillis, les métriques cumulées sont initialisées. Cette affectation est statique pour le décodage de turbocodes sans fenêtrage. Elle respecte dans ce cas (4.15) lorsque l'état initial du treillis est connu. Le décodage fenêtré et parallélisé de turbocodes nécessite de conserver les métriques cumulées de l'itération précédente avec la technique de transfert de métriques *Next Iteration Initialization* [95] introduite dans la section 3.3.1 du chapitre 3.

$$\mathtt{MAC}_0(s) = \mathtt{MAC}_K(s) = \begin{cases} C^{te} & \text{si } s = 0 \\ 0 & \text{sinon} \end{cases} \tag{4.15}$$

Le décodage des codes QC-LDPC ne nécessite pas d'utiliser cette technique. Les métriques sont initialisées en fonction de la parité des états à l'aide d'une constante prédéfinie C^{te}. En effet, chaque *treillis de contrainte de parité* est initialisé et terminé à son état de référence 0. le *treillis de contraintes multiples* regroupe verticalement plusieurs treillis, ce qui revient à débuter les treillis sur les états pairs. Cette étape est résumée par la fonction (4.16).

$$\begin{array}{rcl} \texttt{MAC}_0(2.s) = \texttt{MAC}_{d_c}(2.s) & = & C^{te} \\ \texttt{MAC}_0(2.s+1) = \texttt{MAC}_{d_c}(2.s+1) & = & 0 \end{array} \tag{4.16}$$

4.2.2 Architecture de décodage générique multi-cœurs

Dans la section précédente, une architecture de décodage a été mise au point pour permettre le décodage à la volée de mots de codes compatibles avec le standard 3GPP LTE ou IEEE 802.11n. Cette première architecture ne bénéficie cependant ni des opportunités de parallélisme offert par les codes 3GPP LTE, ni de celui des codes QC-LDPC. En effet, une propriété des codes LDPC Quasi-Cyclique est de pouvoir organiser la mise à jour d'un ensemble de z équations indépendantes sans perte de performances, or l'architecture à un cœur SISO ne met à jour que **b** équations en parallèle.

4.2.2.1 Adaptation de l'architecture en fonction des codes

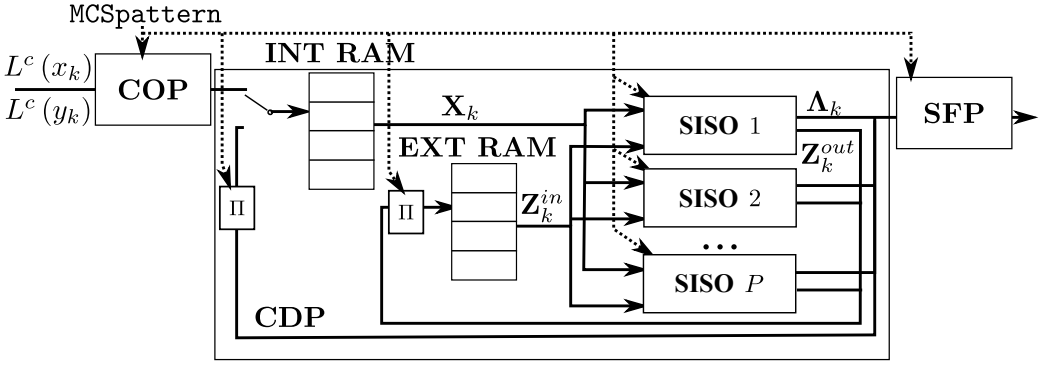


FIGURE 4.11: Architecture de décodage à P cœurs SISO

Une architecture bénéficiant d'un plus grand nombre de cœurs est envisageable pour le traitement des codes QC-LDPC. Pour cela, on définit un nombre maximal de $P = \lceil \frac{z}{b} \rceil$ cœurs pouvant être utilisés conjointement.

Du côté du turbocode, la technique de fenêtrage est couramment employée pour réduire la taille des mémoires du cœur SISO, mais également pour paralléliser le décodage du treillis en découpant celui-ci en tronçons. L'entrelacement doit être étudié avant d'opérer une découpe en P tronçons du treillis.

En fonction du scénario de convergence choisi, il est possible d'établir deux stratégies d'architecture. Une première stratégie consiste à implémenter une architecture contenant le nombre maximal de cœurs SISO pour les standards gérés. Ainsi, on choisit d'implémenter $P = \max(P^{\text{code}_1}, P^{\text{code}_2})$. Cependant, cette architecture mobilise de nombreuses ressources matérielles non exploitées par le décodeur dans l'un ou l'autre cas d'usage. Une seconde stratégie consiste à définir le nombre de cœurs SISO permettant de limiter les ressources non exploitées pour chaque cas d'usage. C'est cette seconde stratégie qui sera détaillée dans cette section. Dans le cas d'une architecture compatible avec les standards 3GPP LTE et IEEE 802.11n, une architecture à 8 cœurs permet de réduire les ressources matérielles non exploitées.

4.2.2.2 Adaptation du banc de mémoires pour P cœurs SISO

On suppose dans cette section que les entrelacement des turbocodes et des codes QC-LDPC sont sans conflit d'accès pour le nombre de cœurs SISO P choisi. Cette propriété est vérifiée en suivant le processus décrit dans le chapitre 3 (section 3.3.2). On cherche alors à caractériser les bancs de mémoire et à ajuster leurs dimensions en fonction des codes à traiter. Chaque cœur SISO requiert $\mathbf{b}$ informations stockées dans les bancs de mémoire **INT RAM** et dans les bancs de mémoires **EXT RAM** afin de traiter une section de treillis. Cette information est requise sur un même cycle d'horloge et nécessite donc de scinder ces deux mémoires sur $\mathbf{b}$ RAM par cœur de décodage. Pour P processeurs indépendants, les bancs de mémoires requièrent alors $\mathbf{b}.P$ RAM chacun. La contrainte de profondeur doit être rediscutée. Pour définir cette caractéristique, cette section propose de séparer l'étude des contraintes liées au code QC-LDPC et au turbocode.

Pour les turbocodes, on scinde le treillis équivalent (naturel ou entrelacé) en P tronçons de K/P sections. Dans ce cas, l'adresse relative à un cœur p pour une section k vaut alors $p = \lceil \frac{k}{K/P} \rceil$ et $\texttt{ADDR} = (k)_{[K/P]}$. De ce fait, la profondeur maximale de la RAM est donnée par le nombre de sections de treillis par tronçon, soit K/P.

La découpe du treillis suivant le code QC-LDPC est relative au treillis défini dans le chapitre 3. Pour un sous-groupe d'équations indépendantes $\mathcal{C}^{\mathbf{m}}$, il existe exactement $\lceil \frac{z}{b} \rceil$ treillis de contraintes multiples dont le décodage est indépendant. De ce fait, un cœur de processeur SISO peut traiter ces treillis en série. Chaque cœur a à charge l'équivalent de $\lceil \frac{z}{b} \rceil / P.d_c^{\mathbf{m}}$ treillis. Au total, chaque cœur traite $\lceil \frac{z}{\mathbf{b}.P} \rceil . \sum_{\mathbf{m}}(d_c^{\mathbf{m}})$ sections par itération, ce qui correspond alors à la profondeur de RAM.

La Table 4.4 évalue la profondeur des mémoires pour le traitement d'une itération pour une architecture mêlant plusieurs cœurs SISO. Cependant, l'usage de plusieurs processeurs en parallèle nécessite de multiplier les bancs mémoires d'autant.

4.2.2.3 Débits de décodage

Le débit obtenu par les architectures de décodage précédemment décrites s'obtient à partir de l'équation (3.36) approchée au chapitre 3 corrigée par le délai de

P	3GPP LTE	IEEE 802.11n
$P = 1$	6144	1848
$P = 2$	3072	968
$P = 4$	1536	528
$P = 8$	768	264
$P = 16$	384	176
$P = 32$	192	88

TABLE 4.4: Profondeur des mémoires en fonction du nombre de cœurs SISO pour traiter tous les MCS des standards 3GPP LTE et IEEE 802.11n pour $\mathbf{b} = 4$

latence entre le décodage des différents treillis noté δ_t et le délai pour gérer les différents modules de décodage noté δ_{it}. Ce débit dépend également de la répartition des tailles de fenêtre w, qui sont fixées à 32 pour le décodage de codes 3GPP LTE et qui dépendent de la répartition des degrés de parité des matrices QC-LDPC. La relation (4.17) fournit alors le débit de décodage de l'architecture en fonction de ces paramètres pour un cœur SISO en *Papillon*.

$$D = K.\frac{1}{N_{it}.\left(\sum_{m=1}^{M}\left(2.\left\lfloor\frac{w(m)}{2}\right\rfloor + \delta_T(m, MCS)\right) + \delta_{it}\right)} \times f_{clk} \tag{4.17}$$

Dans cette équation, M désigne le nombre de treillis du code par itération de décodage. Ce paramètre dépend du nombre de processeur SISO en parallèle P. Pour le cas de décodage de codes 3GPP LTE, l'ensemble du décodage est effectué sur $M = 2 \times P$ treillis. Dans ce cas, chaque processeur SISO traite une taille de fenêtre équivalent à $w = T_l/P$. La taille de la fenêtre pour le décodage de codes QC-LDPC vérifie $w = d_c^{\mathbf{m}}$. Le nombre de treillis pour ce cas d'usage vérifie (4.18).

$$M = \left\lceil \left\lceil \frac{z}{b} \right\rceil / P \right\rceil \times \frac{N-K}{z} \tag{4.18}$$

4.3 Prototypage des architectures

Les architectures de décodage présentées dans la section 4.2 ont été synthétisées pour le cas de convergence de standards 3GPP LTE et IEEE 802.11n. L'implantation des structures à 1 puis 8 cœurs sur une carte FPGA permet de vérifier la preuve de concept de ce projet. Les métriques de performances matérielles sont présentées dans cette section. L'architecture de décodage à 8 cœurs SISO est également proposée pour une synthèse sur une technologie ASIC TSMC 65 nm. Les résultats obtenus pour cette cible sont comparés à ceux présentés dans la littérature.

4.3.1 Portabilité des architectures sur cibles FPGA

4.3.1.1 Contraintes matérielles et protocole expérimental

La plateforme de test utilisée est représentée sur la FIGURE 4.12. Elle a été développée au sein du laboratoire d'Orange-Labs (Cesson-Sévigné, FRANCE) par Marc LANOISELEE. Cette carte reprend deux produits FPGA de la famille Virtex6, le circuit XC6VLX365T et le circuit XC6VSX315T.

FIGURE 4.12: Photographie de la plateforme de test FPGA

Les architectures de décodage ont été intégrées sur une plateforme de test afin d'en vérifier les performances de décodage et de reconfiguration. Cette plateforme est schématisée sur la FIGURE 4.13. Le décodeur interagit avec la plateforme de test à l'aide d'un signal permettant l'envoi d'un message. Un générateur de contexte sélectionne un contexte de Modulation et Schéma de Codage (MCS) suivant un standard, une taille de code et un rendement. Ce processus enclenche alors la lecture d'un mot de code équivalent stocké dans une mémoire dédiée et modulé suivant la modulation BPSK.

Afin de tester le décodeur, un générateur de bruit a été ajouté permettant d'obtenir un bruit blanc additif gaussien suivant la loi de probabilité Normale $\mathcal{N}(0,1)$. Le bruit est ensuite pondéré en fonction du rapport signal à bruit testé et du rendement de codage du code émis. Celui-ci est ajouté à la sortie du modulateur. La somme est ensuite démodulée et les Log-Rapports de Vraisemblance sont fournis en même temps que le contexte de décodage à l'architecture de décodage. Une fois le décodage effectué, les décisions systématiques sont comparées au message émis.

En pratique, le décodage de mots de code 3GPP LTE de taille $K = 256$ et $R = 1/3$ et le décodage de mots de code IEEE 802.11n $N = 648$ et $R = 1/2$ sont testés aléatoirement. Le taux d'erreur binaire est fourni en fonction du mot de code et en fonction du SNR.

L'ensemble de la plateforme de test a été intégré sur le composant Virtex6 XC6VLX365T dont les caractéristiques sont établies dans la TABLE 4.5. Ce com-

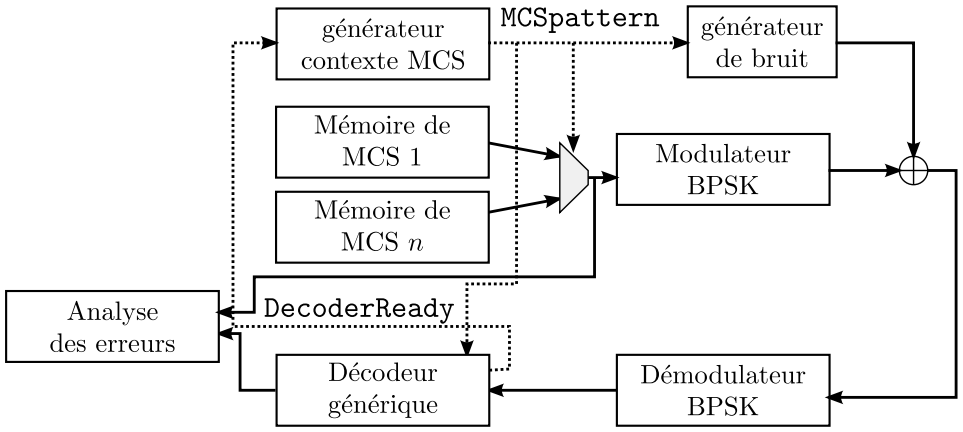


FIGURE 4.13: Plateforme de test du prototype FPGA

posant est surdimensionné pour les usages de cette thèse, aussi les architectures dédiées à une cible FPGA ont été travaillées de façon à réduire les chemins critiques et d'améliorer les fréquences d'utilisation et les débits de décodage.

Caractéristiques	**Données**
Cellules logiques	364,032
Slices	56,880
Slices DSP48E1	576
Blocks RAMs (36 Kb)	416
Input/Output pins	720

TABLE 4.5: Caractéristiques techniques de la technologie FPGA Virtex6 XC6VLX365T

La conception de la plateforme de test étant récente, il n'existe actuellement pas d'interface permettant la récupération des sorties de décodage ni du calcul d'erreurs. De ce fait, un espionnage des signaux aux moyens de l'outil Chipscope ISE est nécessaire pour récupérer les données de performance des architectures testées.

4.3.1.2 Intégration d'une architecture de décodage à un cœur

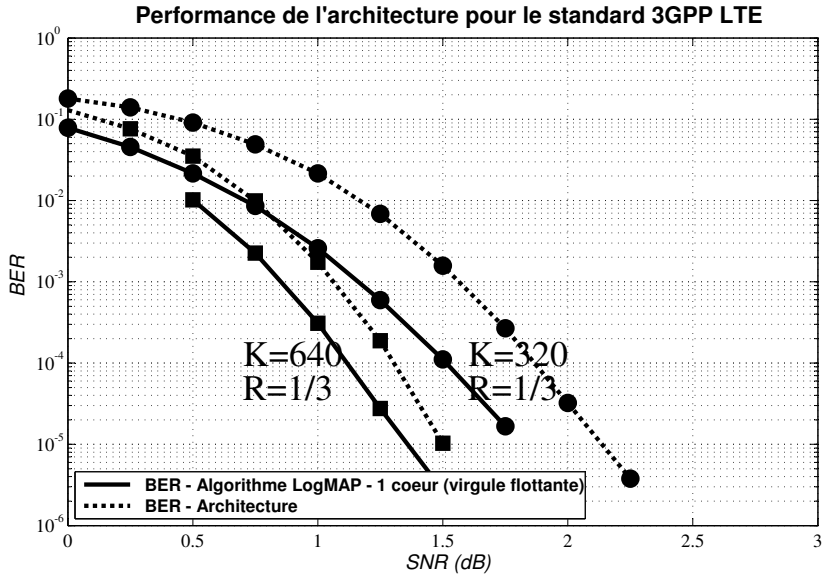
L'intégration d'une architecture à un cœur SISO sur une cible FPGA constitue une première application de ce décodeur conjoint. Cette architecture doit répondre aux problématiques de conflits d'accès des mémoires pour le décodage de codes QC-LDPC. Il s'agit également d'une architecture basique qui ne cherche à optimiser ni les débits, ni la complexité. Cependant, ce type d'architecture est adaptable pour de nombreux cas de convergences entre turbocode et code QC-LDPC pour peu que l'on adapte le cœur de décodage SISO et l'entrelacement dédié en fonction du turbocode et que l'on modifie les paramètres d'entrelacement pour le décodage d'autres matrices QC-LDPC.

La cible FPGA est capable de décoder aléatoirement des mots de codes avec une fréquence d'utilisation de 155 MHz. La faible fréquence d'utilisation par rapport à la fréquence maximale d'utilisation des cœurs SISO est due au calcul de l'entrelacement dédié au décodage turbo. La TABLE 4.6 résume le rapport de synthèse de cette architecture. Cette synthèse montre une forte mutualisation des ressources entre la structure générique et deux structures compatibles à un standard 3GPP LTE ou IEEE 802.11n. En effet, le nombre de LUT est augmenté de 4,2 % (resp. 5,3 %) sur la structure conjointe par rapport aux solutions de décodage d'un standard 3GPP LTE (resp. IEEE 802.11n) seul.

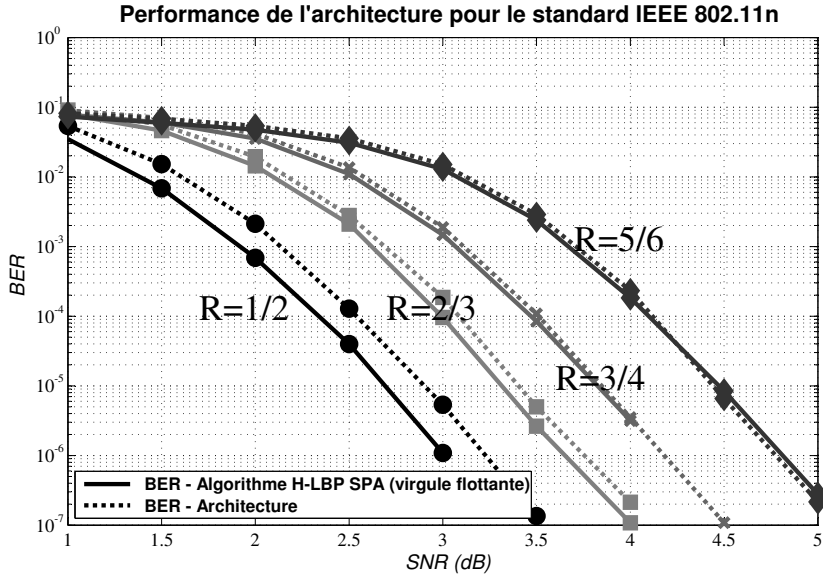
	Décodeur conjoint	**Décodeur LTE**	**Décodeur 802.11n**
Utilisation de slices logiques			
Slice Registers	20645 (4 %)	20385 (4 %)	20265 (4 %)
Slice LUTs	44711 (19 %)	42899 (18 %)	42437 (18 %)
utilisé comme mémoire	7123	7091	7059
Distribution des slices logiques			
LUT Flip Flop pairs	45482	43668	43103
Block RAM			
Block RAM	26 (6 %)	26 (6 %)	26 (6 %)
Fréquence			
Fréquence	155 MHz	155 MHz	165 MHz

TABLE 4.6: Résultat de synthèse sur Virtex 6 (XC6VLX365T) de l'architecture de décodage à un cœur SISO compatible avec les standards 3GPP LTE et IEEE 802.11n

Pour cette architecture, une dynamique interne de 16 bits a été appliquée afin de vérifier la compatibilité de l'architecture avec les courbes théoriques. Les RAM ont été dimensionnées afin de décoder des mots de taille maximale $K = 4096$. Cette caractéristique comprend donc le décodage des 12 matrices QC-LDPC relatives au standard IEEE 802.11n et les mots de code 3GPP LTE les plus courts. Une amélioration du dimensionnement est nécessaire pour une compatibilité de décodage des mots de codes 3GPP LTE de taille supérieure. L'ordonnancement des opérations de réception, de décodage et de sortie de code suivent le cheminement défini par la FIGURE 4.8. Dans le cadre d'un décodage de turbocode 3GPP LTE, le nombre d'itérations a été fixé à 6 et à 10 itérations pour le décodage de codes QC-LDPC. Les performances de décodage obtenues suivant les simulations sur un simulateur codé en C et à travers l'architecture sont observables sur la FIGURE 4.14. Le cœur de décodage SISO applique l'échelonnage des informations de décisions et des informations extrinsèques, ce qui permet d'obtenir des dégradations de performance BER inférieure à 0.20 dB par rapport au décodage des mêmes codes QC-LDPC pour un algorithme H-LBP SPA avec le même nombre d'itérations, ainsi que par rapport à l'algorithme LogMAP sur treillis complet pour les turbocodes du standard 3GPP



(a)



(b)

FIGURE 4.14: Courbe du taux d'erreur atteint par l'architecture pour des mots de codes 3GPP LTE - 6 itérations (a) et IEEE 802.11n - 9 itérations (b) comparée aux algorithmes optimaux

LTE.

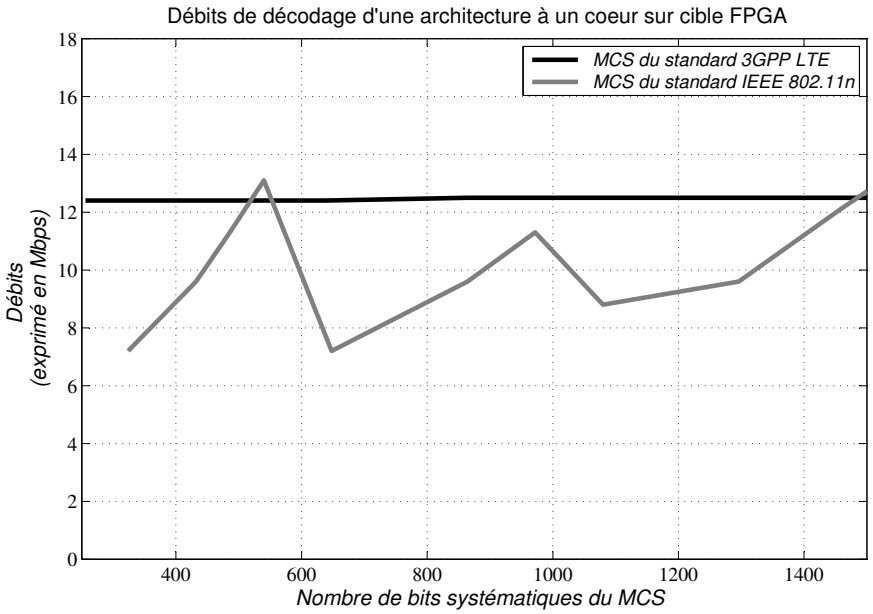


FIGURE 4.15: Débit de décodage pour l'architecture à un cœur et pour une fréquence d'utilisation de 150 MHz

Une architecture à un cœur SISO permet de vérifier la flexibilité du contexte de décodage à la volée. Pour réaliser cette architecture, des mots de code sont émis avec un changement de contexte tous les deux mots reçus. La FIGURE 4.15 montre les débits de décodage attendus par cette architecture pour plusieurs codes dans un contexte sans reconfiguration.

Dans le cadre du décodage de MCS IEEE 802.11n, le rendement de codage impacte le débit de décodage. Cette différence s'explique par un nombre d'itérations fixé pour tout code QC-LDPC et par la taille de treillis dépendant principalement du nombre de nœuds de la matrice associée. Pour le standard IEEE 802.11n, le nombre de nœuds est du même ordre pour les mêmes tailles de mots de code. Ainsi, à itération constante, la latence de décodage est du même ordre pour le décodage de ces MCS, mais plus d'informations sont décodées pour des rendements élevés, ce qui explique l'allure de la courbe.

Le standard 3GPP LTE permet de bénéficier de rendements de codage adaptés en poinçonnant le mot de code. Les différents rendements ne modifient pas la structure du treillis de code convolutif constituant, mais remplace certaines informations du canal par des éléments neutres. Pour un même nombre d'itération, la latence de décodage ne varie donc pas. Ainsi, les débits de décodage sont indépendants du rendement du code. Par contre, le nombre de sections de treillis décodées est proportionel au nombre d'informations systématiques. La latence de décodage est donc pratiquement proportionnelle à la taille du message d'information K, ce qui explique que le débit de décodage soit faiblement influencé par K.

Pour un usage aléatoire et identiquement réparti des deux codes testés, l'architecture tourne avec une performance de décodage de 7.00 Mbps. En pondérant ces débits en fonction des données utiles fournies par chaque standard, le débit effectif du décodeur est de 3.09 Mbps de données 3GPP LTE et 3.91 Mbps de données IEEE 802.11n. Ces résultats ont été validés par calcul et par mesures sur simulations VHDL.

La reconfiguration du cœur SISO entraîne également un coût sur les débits de décodage. Pour le plus petit mot de code du standard 3GPP LTE, le débit sans reconfiguration entre chaque mot de code est de 10,9 Mbps sur cette structure pour une latence de 552 cycles d'horloge. La reconfiguration pour le décodage de cette configuration est de 5 cycles d'horloges, ce qui réduit les débits à 10,8 Mbps, soit une perte de 0,9 %. Cependant, ce coût est infime pour l'adaptation au standard IEEE 802.11n. En effet, le débit de décodage atteint est de 7,23 Mbps pour une latence de décodage du MCS le plus court de 6726 cycles d'horloge sans reconfiguration. Cette latence augmente de 7 cycles d'horloge en prenant en compte la reconfiguration ce qui réduit le débit à 7,22 Mbps, soit une dégradation de moins de 0,01 %.

Ces débits sont très faibles par rapport à la littérature. Nous avons donc également implanté un décodeur avec plusieurs cœurs SISO pour améliorer ces débits de décodage.

4.3.1.3 Intégration d'une architecture de décodage à 8 cœurs

Suite à la réalisation d'une architecture de décodage à un cœur SISO, l'intégration s'est poursuivie par une architecture de décodage à plusieurs cœurs SISO afin d'obtenir une preuve de concept d'une architecture de décodage générique et flexible aux débits plus compétitifs. La stratégie de cette architecture a été de réaliser un décodeur compatible pour les MCS 3GPP LTE et IEEE 802.11n intégrant 8 cœurs SISO. De ce fait, les débits ne sont optimaux pour aucune des applications, mais la complexité de l'architecture est réduite pour les deux cas d'application.

Cette nouvelle architecture est synthétisée sur la même carte FPGA Virtex6 avec une fréquence d'horloge de 140 MHz. La dynamique des métriques est conservée sur 16 bits. L'architecture est dimensionnée pour décoder des mots contenant au plus 4096 sections de treillis, ce qui comprend l'ensemble des mots de codes du standard IEEE 802.11n et les mots de codes les plus petits du standard 3GPP LTE. Les données de synthèse sont fournies dans la Table 4.7. Pour cette architecture, les paramètres tels que le nombre d'itérations sont conservés par rapport à l'architecture à un cœur SISO. Les performances BER de cette architecture sont les mêmes que pour une architecture à un cœur SISO pour le décodage des MCS IEEE 802.11n ce qui correspond aux performances attendues. L'architecture suit la technique de fenêtrage et la transition NII définie précédemment. Pour cette architecture, les performances sont bit-true, et correspondent donc aux performances fournies par le simulateur C. L'architecture intégrée sur cette cible permet d'atteindre 93 Mbps pour le décodage 3GPP LTE pour le décodage de mots de code de rendement 1/3 et 101 Mbps pour le décodage IEEE 802.11n du MCS $N = 1944$ et $R = 5/6$. Le

débit de décodage est pratiquement multiplié par 8 dans chaque cas d'usage.

[107] propose une architecture de décodage pour un seul mot de code du standard IEEE 802.11n. Les résultats obtenus par l'architecture pour 5 itérations de décodage suivant l'algorithme par inondation et mise à jour Min-Sum atteint 38,3 Mbps sur cible FPGA Virtex2 à 194 MHz. L'algorithme sélectionné converge deux fois moins vite que l'algorithme de notre architecture. De ce fait, en ramenant cette structure à des performances équivalentes, l'architecture de décodage présentée est plus compétitive par rapport à certains choix d'architectures dédiées de la littérature et implantées sur cible FPGA.

[108] propose une architecture de décodage de codes 3GPP LTE sur une cible FPGA non précisée. Pour une fréquence d'utilisation de 102 MHz, cette structure atteint 347 Mbps pour 2 itérations de décodage. La structure de décodage suivant le radix-4 a été privilégiée avec un décodage Max-LogMAP. Le nombre d'itérations a été fixé pour répondre aux exigences de débits du standard 3GPP LTE. Ramenée aux performances de taux d'erreur binaire (6 itérations), cette structure obtient un débit de 115 Mbps. La structure multistandard proposée présente donc des débits dégradés par rapport à une architecture dédiée.

	Architecture 8 cœurs	**Ressources de la cible**
Utilisation de slices logiques		
Slice Registers	43781	455040 (9 %)
Slice LUTs	82663	227520 (36 %)
utilisé comme logic	82597	
utilisé comme mémoire	66	
Distribution des slices logiques		
LUT Flip Flop pairs	96491	
Block RAM/FIFO		
Block RAM	120	416 (28 %)
Fréquence		
Fréquence	140 MHz	

TABLE 4.7: Synthèse de l'architecture de décodage à 8 cœurs SISO compatibles avec les standards 3GPP LTE et IEEE 802.11n pour une architecture avec ordonnancement du treillis en *Papillon*, sur une carte Virtex 6 (XC6VLX365T)

4.3.1.4 Déroulement du processus de décodage

Les deux architectures précédentes ont été intégrées sur une carte FPGA. Le fonctionnement de celles-ci a été mesuré directement sur la cible FPGA à l'aide de l'outil Chipscope. Nous revenons sur le déroulement du processus de décodage et sur les interactions entre les différents blocs. Cette approche permet de visualiser la reconfigurabilité du décodeur entre différents contextes de décodage.

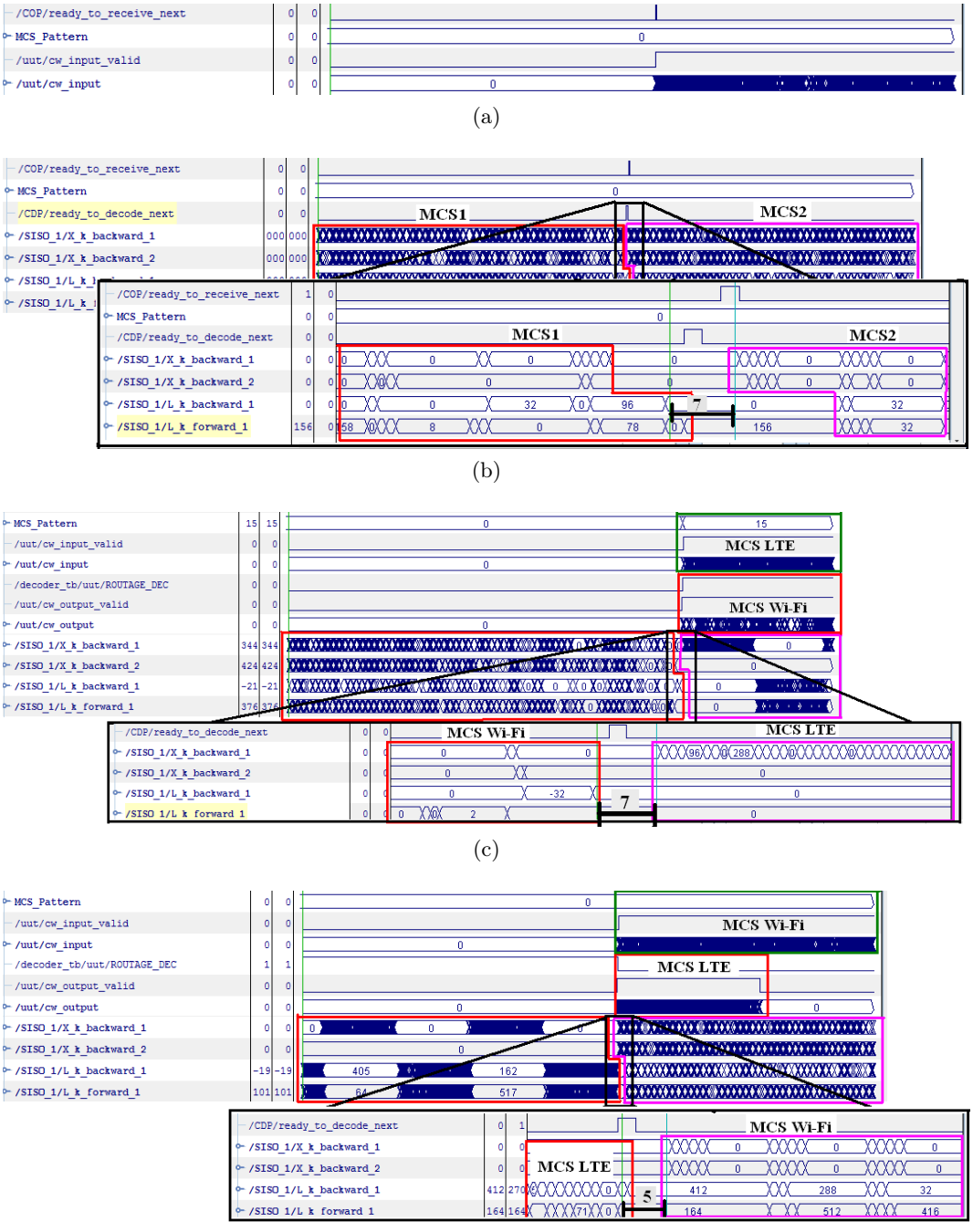


FIGURE 4.16: Déroulement du processus de décodage flexible entre deux contextes des standards 3GPP LTE et le standard IEEE 802.11n

Le module de réception des messages externes **COP** communique avec la plateforme de test au moyen du signal `Ready_to_receive_next`. La plateforme envoie un message à décoder dès que celui-ci est apte à traiter un message. Dans ce cas, le jeton `MCS_pattern` permet au décodeur de connaître les caractéristiques du code de référant. Le message est ainsi enregistré dans les mémoires attribuées en fonction du modèle défini. La FIGURE 4.16a montre l'intégration de cette étape analysé grâce à l'outil Chipscope.

Indépendamment de ce processus, le module de décodage **CDP** interroge le processeur **COP** par un signal `ready_to_decode_next`. Si ce module contient un message à décoder, le processus de décodage est lancé. Lorsque le le module de décodage **CDP** est disponible pour décoder un message, il interroge le processeur **COP** par le signal `ready_to_decode_next`. Le contexte de décodage est alors fourni au module de décodage. Le **COP** lière dans ce cas l'espace requis par le message récemment décodé pour réceptionner un autre message. La transition entre deux MCS par le **CDP** dépend du contexte de décodage. La FIGURE 4.16b représente une transition de décodage entre deux messages codés suivant le même MCS du standard IEEE 802.11n. Dans ce cas de transition, 7 cycles d'horloge ont été mesurés pour initialiser le bloc de décodage.

Le module de sortie **SFP** est prévenu de la fin du décodage par le module **CDP** avec le signal `flush_en` et lance l'extraction des informations systématiques *a posteriori* (signal `cw_out_valid` et `cw_output` en indiquant le standard auquel il appartient (signal `ROUTAGE_DEC`). Le module **SFP** annonce également sa disponibilité au module processeur **COP** qui lui donne un autre mot à décoder (`ready_to_decode_next`).

Le contexte de décodage est adapté à chaque mot de code. La FIGURE 4.16c et la FIGURE 4.16d mettent en évidence le décodage de messages suivant deux contextes différents. Sur la FIGURE 4.16c le décodage d'un MCS du standard 3GPP LTE succède au décodage d'un MCS du standard IEEE 802.11n. La reconfiguration du processeur de décodage s'effectue à la volée. Pour ce cas de transition, 7 cycles d'horloge permettent la signalisation de la fin du décodage et le changement de contexte pour une adaptation au standard suivant. Pour une transition inverse représentée sur la FIGURE 4.16d, 5 cycles d'horloge sont nécessaires. Cette latence représente le délai entre la dernière sortie des modules de décodage SISO et le début du décodage SISO du mot de code suivant. La différence de latence entre ces deux cas de transition provient de la différence de latence de décodage des deux treillis de ces codes.

4.3.2 Portabilité des architectures sur cibles ASIC

4.3.2.1 Adaptation à la technologie ASIC TSMC 65 nm

Le choix d'une synthèse sur une cible ASIC (*Application-Specific Integrated Circuit*) de la technologie 65 nm permet d'obtenir des résultats d'intégration comparables avec de nombreuses architectures présentes dans la littérature. Cependant,

le changement de technologie entraîne des modifications de l'architecture pour répondre à des problématiques différentes. Contrairement à une cible FPGA, qui contient généralement des multiplieurs et des blocs RAM, l'ASIC est un circuit intégré complètement dédié à l'architecture voulue et ne comportant que les éléments et opérateurs strictement nécessaires. Dans cette partie, les modifications apportées à l'architecture et les résultats de synthèse sur cible ASIC sont fournis. La réalisation matérielle (layout, vérifications, opérations de fonderie et test) n'a pas été effectuée.

La cible de synthèse s'est basée sur les librairies de **Taiwan Semiconductor Manufacturing Company** (TSMC). L'architecture a été synthétisée sur une technologie de 65 nm. Les intégrations du moment privilégient cette dimension de gravure minimale. Pour une architecture ASIC, la fréquence de fonctionnement et le débit utile de décodage sont des métriques qui permettent de qualifier le composant résultant du point de vue de ses performances brutes. À cela s'ajoute une autre métrique d'un intérêt essentiellement économique, et qui est la surface de silicium nécessaire à la réalisation de la fonction. Cette dernière information détermine dans une certaine mesure le rendement de fabrication, et *in fine* le coût de chaque circuit. De ce fait, l'architecture définie sur une cible ASIC doit vérifier une complexité mesurée en surface de silicium qui soit compétitive par rapport aux architectures concurrentes.

Pour répondre en premier lieu à cette problématique, les blocs RAM sont synthétisés en externe de manière à mesurer leurs impacts sur la surface globale. La Table 4.8 donne quelques synthèses de blocs RAM pour la technologie TSMC ASIC 65 nm réalisées avec l'outil ARTISAN fourni par ARM. Ce tableau montre l'impact du choix de quantification et du choix de la technologie RAM utilisée. Les surfaces de silicium affectées aux RAM double-ports (DPRAM) sont de deux à quatre fois plus importantes que des RAM simple port (SPRAM). De plus, ces RAM fonctionnent avec une fréquence maximale de 3 GHz. De ce fait, l'architecture a été adaptée pour utiliser des RAM simple port avec une fréquence double plutôt que de conserver les RAM double-port utilisées pour la cible FPGA.

Nombre de mots	**Nombre de bits**	**Surface** *DPRAM* (en mm^2)	**Surface** *SPRAM* (en mm^2)
512	32	$8,40.10^{-2}$	$3,91.10^{-2}$
512	24	$6,77.10^{-2}$	$3,02.10^{-2}$
512	22	$3,30.10^{-2}$	$2,79.10^{-2}$
512	20	$5,95.10^{-2}$	$2,57.10^{-2}$
512	16	$5,13.10^{-2}$	$2,13.10^{-2}$
512	10	$3,91.10^{-2}$	$1,46.10^{-2}$
512	9	$3,70.10^{-2}$	$1,34.10^{-2}$
512	8	$3,50.10^{-2}$	$1,23.10^{-2}$

Table 4.8: Tableau de synthèse des SPRAM et DPRAM sur une technologie TSMC ASIC 65 nm

4.3.2.2 Synthèse d'une architecture de décodage multi-cœurs sur cible ASIC

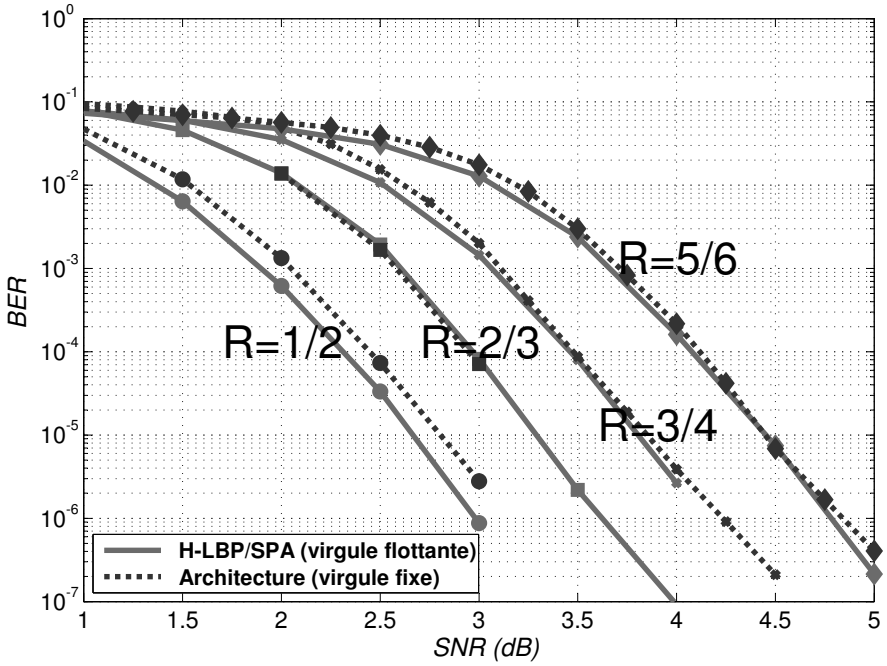


FIGURE 4.17: Performances de décodage BER pour des mots de codes IEEE 802.11n comparées aux algorithmes optimaux avec la quantification ASIC

L'architecture de décodage a été revue pour modifier les mémoires en SPRAM. L'architecture a également été optimisée avec une modification des facteurs d'échelonnage pour le décodage QC-LDPC. La quantification a été réduite afin de diminuer la complexité de décodage mais également de comparer cette architecture avec les architectures de la littérature. De ce fait, la sortie du démodulateur est quantifiée sur 6 bits avec 5 bits pour coder la valeur entière et 1 pour la valeur décimale. À l'intérieur du décodeur, les informations de décision sont codées sur 10 bits (Q6.4) et les valeurs extrinsèques sur 8 bits (Q5.3). De ce fait, les bancs de mémoires LLR sont de profondeur 512 et de longueurs 10, les deux bancs extrinsèques de longueur 12. Cette nouvelle quantification entraîne des dégradations de performance pour le décodage de codes QC-LDPC. Le plancher d'erreur est rehaussé du fait de la perte d'information sur la valeur décimale de la décision. Pour contrer cet effet, le facteur d'échelonnage a été modifié pour prendre en compte le couple $(0.75, 1.33)$. Les performances BER pour cette nouvelle quantification sont données sur la FIGURE 4.17. La modification du paramètre d'échelonnage permet de conserver des résultats BER de décodage moins dégradés par rapport aux algorithmes optimaux en virgules flottantes. Cependant, pour une matrice de rendement 1/2, ce choix

montre une dégradation de performance de 0.2 dB qui dégrade le décodage de cette matrice. Le choix de quantification reste intéressant pour les autres rendements de codage.

L'architecture a été synthétisée en ciblant une fréquence d'horloge de 500 MHz. Les SPRAM fonctionnent avec une fréquence maximale de 3 GHz, ce qui permet d'envisager cette structure avec plusieurs accès mémoire pendant le même cycle d'horloge à 500 MHz. La Table 4.9 fournit les résultats de synthèse pour cette architecture. Avec 8 cœurs de processeur, cette architecture permet d'atteindre des débits utiles de 330 Mbps pour le décodage 3GPP LTE et 410 Mbps pour le décodage QC-LDPC.

Cette synthèse montre la faisabilité de l'architecture sur une structure ASIC avec une fréquence d'utilisation de 500 MHz. Par extension, une architecture contenant 32 cœurs SISO permettrait d'atteindre des débits utiles de 1,07 Gbps pour le décodage des MCS IEEE 802.11n et de 1,3 Gbps pour des MCS 3GPP LTE.

Fréquence	500 MHz
Surface combinatoire	$5{,}20.10^5$ μm^2
Surface non combinatoire	$3{,}72.10^5$ μm^2
Surface mémoire	$8{,}47.10^5$ μm^2
Surface totale	1,70 mm^2

Table 4.9: Résultat de la synthèse de l'architecture de décodage à 8 cœurs sur une cible ASIC 65 nm

4.3.3 Comparaison de l'approche présentée avec la littérature

Différentes architectures de décodage sur technologie ASIC sont envisagées dans la littérature. Dans cette partie, nous revenons sur certaines d'entre elles afin de comparer cette architecture de décodage avec d'autres choix.

La Table 4.10 reprend les performances architecturales obtenues dans ce travail et les compare à quelques solutions dédiées ou multistandard de la littérature.

[70] propose une architecture de décodage dédiée au standard 3GPP LTE sur une technologie ASIC de 130 nm. Cette architecture cherche à répondre aux débits de décodage du standard LTE de 326 Mbps tout en limitant la consommation énergétique du décodeur. [70] sélectionne une architecture à 8 cœurs de décodage SISO traitant des treillis transformés suivant la technique du Radix-4. Ce choix permet d'améliorer les débits de décodage au prix d'un surcoût de surface de silicium. Ce décodeur favorise 5.5 itérations contre les 6 itérations fixées pour notre structure de décodage. Le rapport de débit par surface, qui est dans ce cas ramené à 6 itérations pour une performance équivalente et à une technologie ASIC de 65 nm rend le rapport de décodage 4 fois plus efficace que notre structure. Cette architecture montre donc une forte performance matérielle pour une architecture compatible à un standard unique.

Références	[70]	[57]	[68]	[68]	[109]	[69]	Ce projet
Standard	LTE	802.11n	802.11n	LTE	802.11n LTE	802.11n HSPA	802.11n LTE
N_{it}	5,5	10	12	6	10 5,5	8 8	10 6
Technologie (*en nm*)	130	90	90	90	130	45	65
Surface (*en mm²*)	3,57	1.77	2,05	2,00	2,8	0,9	1,7
Fréquence (*en MHz*)	302	346	500	500	160	150	714
Débit (*en Mbps*)	390	679	640	250	136 104	122 73,4	482 475
Rapport débit par surface [a] (*en Mbps/mm²*)	**802**	**1085**	**1060**	**350**	**388** **272**	**38** **38**	**279** **275**

a. Cette quantité est mise en perspective pour une technologie ASIC TSMC 65 nm, normalisée pour 10 itérations dans le cas IEEE 802.11n et 6 itérations dans le cas 3GPP LTE

TABLE 4.10: Etat de l'art d'architectures ASIC dédiées et d'architectures flexibles

D'autres architectures "simple standard" ont une efficacité moindre que notre prototype. [110] propose une architecture de décodage Radix-2 avec 16 cœurs de décodage SISO sur une technologie CMOS de 130 nm. Cette architecture utilise 17.81 mm^2 de silicium pour une fréquence de fonctionnement de 80 MHz. Cette architecture permet d'atteindre un débit de 160 Mbps pour 8 itérations. Le rapport débit par surface atteint 95 Mbps/mm^2 ramené à une technologie de 65 nm. Notre architecture est dans ce cas deux fois plus efficace tout en proposant une adaptation au décodage de codes QC-LDPC.

[57] propose une architecture de décodage des codes LDPC répondant au standard IEEE 802.11n. Cette architecture favorise l'algorithme de décodage H-LBP - OMS avec des performances de décodage proche de notre choix algorithmique. Cette architecture bénéficie du parallélisme optimal pour le code QC-LDPC (81 décodeurs parallèles) et permet d'obtenir des débits de 679 Mbps sur 1.77 mm^2 de silicium avec une technologie de 90 nm. Le rapport débit par surface atteint dans ce cas 1Gbps/mm^2. Cette structure représente donc une solution très efficace pour un usage dédié à un standard unique.

D'autres architectures dédiées sont moins efficaces pour le décodage de codes QC-LDPC. Ainsi [111] propose une architecture de décodage sur une technologie ASIC de 180 nm qui permet d'atteindre 57 Mbps à 10 itérations de décodage pour le standard IEEE 802.16e. Le rapport débit par surface est alors de 91 Mbps/mm^2 ce qui est moins efficace que notre architecture.

[68] propose d'appliquer le même algorithme pour décoder les codes des standards 3GPP LTE et IEEE 802.11n. L'algorithme LogMAP est appliqué aux treillis associés aux conditions de parité d'un code QC-LDPC. De ce fait, les performances de décodage atteintes s'en trouve améliorées. Cette architecture a été synthétisée sur une technologie ASIC de 90 nm et permet d'obtenir un rapport débit par surface compétitif en comparaison aux premières architectures décrites. Cependant, ces solutions sont dédiées à un standard unique et ne participent pas au projet de reconfiguration mené dans ce travail.

[69] propose une architecture ASIC flexible sur une cible de 45 nm privilégiant une faible consommation énergétique pour décoder les codes correcteurs avancés des standards IEEE 802.11n, IEEE 802.16e, DVB-SH ainsi que 3GPP HSDPA. Pour cela, chaque cas d'usage fait intervenir 8 itérations de décodage avec une dégradation d'implémentation de l'ordre de 0.25 dB sur le FER. Le choix d'une faible consommation énergétique implique alors un faible rapport débit par surface qui atteint 38 Mbps pour chaque usage soit un rapport quatre fois moins efficace que notre architecture.

[109] présente également une architecture ASIC flexible sur une cible de 130 nm compatible avec les codes QC-LDPC des standards IEEE 802.16e et IEEE 802.11n et avec le turbocode du standard 3GPP LTE. Cette architecture privilégie un algorithme Max-LogMAP qui présente une dégradation de l'ordre de 0.25 dB pour le décodage d'un code QC-LDPC de rendement 1/2 pour un BER de 10^{-4} alors que pour ce type de code, notre architecture obtient une dégradation de l'ordre de 0.10 dB pour ce rendement de codage, comme indiqué sur la FIGURE 4.17. Ainsi, bien que cette structure bénéficie d'un rapport débit par surface plus avantageux de l'ordre d'un facteur deux, ce bénéfice s'obtient au prix d'une plus importante dégradation de performances.

Notre architecture de décodage convergeant les standards 3GPP LTE et IEEE 802.11n offre une possibilité de changer de technologie de décodage à la volée. Nous avons paramétré cette architecture afin de favoriser le taux de réutilisation des ressources matérielles entre les différents usages, au prix d'une dégradation notable des débits de décodage. L'algorithme Max-LogMAP a été sélectionné pour encourager cette mutualisation. Cependant, les performances BER obtenues ont été améliorées à l'aide de facteurs d'échelonnage appliqués sur les deux types de codes. De ce fait, bien que cette structure ne soit pas optimale par rapport à d'autres choix disponibles dans la littérature, elle montre des performances BER d'une dégradation inférieure à 0.2 dB pour un BER de 10^4 qui, à itération constante, présente de meilleurs performances que [109]. Les architectures dédiées les plus efficaces présentent un meilleur compromis débit par surface. Cependant, de nombreuses architectures dédiées au décodage de turbocodes comme [110] et de codes QC-LDPC comme [111] sont moins efficaces que notre architecture multistandard. Cette architecture montre donc que les coûts matériels supplémentaires dus à la convergence de plusieurs standards restent acceptables pour obtenir une architecture multistandard.

4.4 Conclusion

Ce chapitre est consacré au prototypage d'architectures de décodage génériques et flexibles pour codes correcteurs d'erreurs avancés. Une étape préalable a structuré un cœur SISO compatible avec plusieurs codes correcteurs selon les treillis définis dans le chapitre 3 avec l'établissement d'un exemple concret pour le cas de convergence 3GPP LTE et IEEE 802.11n. Ce cœur est ensuite incorporé dans diverses architectures de décodage consacrées au cas de convergence des standards 3GPP LTE et IEEE 802.11n. Une architecture à un cœur a été élaborée pour répondre aux conflits dus à la convergence des codes. La perte en ressources matérielles due à la généricité du décodeur reste cependant inférieur à 6 % par rapport à une structure dédiée à un seul standard. Cette architecture s'adapte également à d'autres usages en modifiant les paramètres d'entrelacement, les caractéristiques des RAM et le cœur SISO.

Une seconde architecture permet d'améliorer le parallélisme de décodage en prenant en compte les caractéristiques des codes traités. Cependant, une étude au cas par cas est nécessaire pour définir ce parallélisme. Dans le cadre de ce travail, nous avons favorisé une optimisation des ressources matérielles au détriment des débits de décodage en sélectionnant une architecture à 8 cœurs SISO. D'autres choix favorisant les débits sont envisageables, en sélectionnant une architecture de décodage avec un nombre de cœurs SISO optimal pour chacun des standards décodés. Le degré de mutualisation des ressources matérielles est dans ce cas moins important.

L'architecture de décodage fournit une preuve de concept de la faisabilité d'une architecture de décodage conjoint dont le contexte puisse changer en quelques cycles d'horloge. La reconfiguration entraîne une dégradation des débits de moins de 1 % en fonction du code décodé. Les performances de décodage BER sont compétitives par rapport aux décodeurs courants, avec une dégradation inférieure à 0.20 dB par rapport aux algorithmes optimaux en virgule flottante pour chaque code. Les débits de décodage obtenus sont cependant dégradés, ce qui est dû à un choix de mutualisation poussée des ressources.

L'architecture à 8 cœurs SISO implantée sur ASIC équivaut à une surface de 1.7 mm^2 ce qui reste très compétitif par rapport aux architectures de décodage dédiées. Les concepts exploités tout au long de ce chapitre s'adaptent à d'autres cas de convergence de standards. Le décodage d'autres codes QC-LDPC est envisageable mais modifie le parallélisme du module de décodage **CDP**. Le décodage de turbocodes double-binaires est également possible mais demande une adaptation du cœur SISO pour réduire le chemin critique dû au calcul des métriques cumulées.

Conclusion

Les protocoles de communication standardisés montrent une grande variété de stratégies de codage de canal permet de répondre aux contraintes d'usages comme la qualité de la transmission, des débits importants ou une latence faible. Les protocoles de radiocommunication standardisés privilégient de plus en plus de codes correcteurs d'erreurs aux processus de décodage itératifs afin de répondre aux exigences des usages associés.

Nous avons présenté dans le chapitre 1 les principaux codes correcteurs d'erreurs sélectionnés dans des protocoles de communication récents. Ceux-ci présentent cependant des caractéristiques communes. La plupart des codes LDPC standardisés sont des codes Quasi-Cycliques, ce qui permet des stratégies de décodage similaires à l'ensemble de ces codes. Les turbocodes choisis sont également de même forme. La plupart des codes sont à 8 états, et les codes simples binaires laissent apparaître des possibilités de regroupement des sections de treillis par la technique Radix-4. Ainsi, de nombreuses architectures de décodage mêlant plusieurs codes QC-LDPC ou turbocodes de différents standards ont déjà été étudiées.

Nous nous sommes intéressés à une structure de décodage réunissant les deux familles de code. Pour cela, nous avons évalué la complexité matérielle du décodage des principaux codes en fonction des différents choix d'algorithmes de décodage. En évaluant la complexité de décodage pour atteindre une capacité de correction proche, nous avons pu hiérarchiser ces différents algorithmes. La définition d'un diagramme en toile d'araignée a permis de comparer ces complexités en fonction des principales opérations requises. Ainsi, ce chapitre a proposé de comparer la complexité de différents codes décodés par des algorithmes distincts tout en fixant un seuil de capacité de correction équivalent. Suite à ces études, l'algorithme BCJR a été sélectionné comme un algorithme de décodage conjoint aux codes LDPC et aux turbocodes avec un fort potentiel de mutualisation de ressources.

À partir de ce constat, nous avons étudié la structure en treillis des codes QC-LDPC et des turbocodes dans le chapitre 3. Chacun des codes a été décrit en fonction d'une représentation en treillis commune dont les caractéristiques permettent une mutualisation des ressources de calcul. Les caractéristiques d'entrelacement des codes QC-LDPC ont été étudiées afin de préparer l'accès aux informations stockées en mémoire. Les caractéristiques d'entrelacement des turbocodes ont été également vérifiées pour garantir un décodage en fenêtre des treillis. Enfin, les contraintes matérielles de décodage abordées dans le chapitre 2 ont souligné un conflit de décodage de codes QC-LDPC. Nous avons proposé un ordonnancement de calcul des opérations de décodage pour réduire l'impact de cette contrainte.

Dans le chapitre 4, nous avons sélectionné un cas d'application sur cible matérielle. Plusieurs architectures de décodage liées au cas de convergence 3GPP LTE et IEEE 802.11n ont été réalisées afin de vérifier la faisabilité d'une architecture de décodage flexible. Dans un premier temps, nous avons proposé une architecture

de décodage d'un cœur SISO compatible avec un code QC-LDPC et un turbocode binaire à 8 états. Les opérations séquentielles de ce cœur permettent de réduire la latence entre le traitement de deux treillis de même forme à un cycle d'horloge, une transition entre deux treillis différents de 8 à 9 cycles d'horloge afin de vider tout pipeline. Ce cœur est ensuite intégré sur une structure de décodage pour turbocode du standard 3GPP LTE et codes QC-LDPC du standard IEEE 802.11n. Cette structure a été implantée sur une carte FPGA avec un changement de contexte de décodage entre chaque message reçu. Les débits mesurés sont de 3,9 Mbps pour le standard IEEE 802.11n et 3 Mbps pour le standard 3GPP LTE dans le cas le moins favorable. Cette structure est adaptable à tout type de code QC-LDPC en modifiant la fonction d'entrelacement, et à tout type de turbocodes en modifiant également le cœur SISO. La dégradation maximale due à la reconfiguration réduit les débits d'au plus 1 % sur chaque cas d'utilisation, et le surplus de ressources matérielles est inférieur à 6 % par rapport à une architecture dédié à un seul standard.

Une architecture à 8 cœurs a également été proposée pour augmenter ces débits. Sur une cible FPGA fonctionnant à 140 MHz, l'architecture atteint des débits 93 Mbps pour le décodage de MCS du standard 3GPP LTE et de 101 Mbps pour le standard IEEE 802.11n tout en garantissant un pouvoir de correction inférieur à 0,20 dB par rapport aux algorithmes LogMAP et par propagation de croyance. Cette structure a également été portée sur une cible ASIC nécessitant 1.7 mm^2 fonctionnant à 500 MHz. Dans ce cas, les débits atteignent 475 Mbps (3GPP LTE) et 482 Mbps (IEEE 802.11n). La taille de l'architecture est compétitive pour une intégration ASIC.

Cependant, ce travail amène de nombreuses questions supplémentaires. Bien qu'une architecture de décodage conjointe entre un code QC-LDPC et turbocode double-binaire soit envisagée, elle n'a pas été réalisée pour en vérifier les caractéristiques de débit et de surface. Une étude de ce cas d'application serait nécessaire pour répondre à cette question. Une autre interrogation concerne la convergence de trois codes mêlant des codes QC-LDPC, des turbocodes binaire et double-binaire. La représentation Radix-4 d'un turbocode binaire ne permet pas de conserver la structure de treillis intéressante pour le décodage conjoint de ce code avec un code QC-LDPC. Pour répondre à cette question, un travail supplémentaire est inévitable.

Ce projet de thèse répond au décodage de codes variés sur une même architecture. La capacité de correction, les débits de décodage et les caractéristiques architecturales ont été mesurés, et la capacité de reconfiguration et ses coûts en termes de débit ont été soulignés. Cette étude a été réalisée sur un cas de convergence de décodage d'un turbocode 3GPP LTE et de codes QC-LDPC IEEE 802.11n. Les principes énoncés dans ce mémoire s'adaptent cependant à d'autres utilisations et à d'autres types de convergence de standards dont on peut citer la communication flexible sur réseaux mobile et réseaux sans fils personnel ou la communication des maisons interconnectées. L'apport de cette thèse permet également de mêler des stratégies de codage diversifiées sur un protocole de communication futur ce qui garantirait les critères de qualité de service spécifiques à chacune d'entre elles tout en réduisant les ressources matérielles du récepteur.

Annexe A

Glossaire

A.1 Glossaire

AWGN *Additive White Gaussian Noise*

3GPP *3rd Generation Partnership Project*

3GPP2 *3rd Generation Partnership Project 2*

ACS *Addition Comparison and Selection*

ASIC *Application-Specific Integrated Circuit*

BER *Bit Error Rate*

BP *Belief-Propagation*

BPSK *Binary Phase-Shift keying*

CLB *Configurable Logic Blocks*

CSA *Comparison Selection and addition*

DPRAM *Dual Ported Random Access Memory*

DVB *Digital Video Broadcasting*

FER *Frame Error Rate*

FPGA *Field-Programmable Gate Array*

H-LBP *Horizontal Layered Belief-Propagation*

HPAV *HomePlug AV*

IEEE *Institute of Electrical and Electronics Engineers*

ITU *International Telecommunication Union*

LDPC *Low-Density Parity-Check*

LTE *Long Term Evolution*

MAP *Maximum A Posteriori*

MCS *Modulation Coding Scheme*

ML *Maximum Likelihood*

NII *Next Iteration Initialization*

PLC *Powerline Communication*

PSK *Phase-Shift keying*

QC-LDPC *Quasi-Cyclic Low-Density Parity-Check*

QPSK *Quadratic Phase-Shift keying*
RA *Repeat-Acumulate*
RAM *Random Access Memory*
RAN *Radio Area Network*
SISO *Soft-In, Soft-Out*
SNR *Signal over Noise Ratio*
SOVA *Soft-Output Viterbi Algorithm*
SPA *Sum-Product Algorithm*
SPRAM *Simple Ported Random Access Memory*
TSMC *Taiwan Semiconductor Manufacturing Company*
WLAN *Wireless Local Area Network*
SPA *Sum-Product Algorithm*
λM *λ-Min*
MS *Min-Sum Algorithm*
OMS *Offset-Min-Sum Algorithm*
NMS *Normalized-Min-Sum Algorithm*

A.2 Glossaire relatif à l'architecture présentée

AMP *Module de calcul des probabilités d'états*
BMP *Module de calcul des métriques de branches*
CDP *Module de décodage du mot de code*
COP *Module d'ordonnancement du mot de code*
DEC *Module de calcul de décision*
SFP *Module d'extraction des données systématiques*
SISO *Soft-In, Soft-Out*
ACC RAM *RAM de stockage des informations cumulées du cœur SISO*
EXT RAM *RAM de stockage des informations extrinsèques*
INT RAM *RAM de stockage des informations intrinsèques*
SYST RAM *RAM de stockage des informations systématiques du cœur SISO*

Annexe B

Notations

B.1 Notations mathématiques

j	*Nombre imaginaire vérifiant l'équation* $j^2 = -1$
$\|a\|$	*Fonction renvoyant la valeur absolue de* a
$\left\lfloor \frac{a}{b} \right\rfloor$	*Fonction renvoyant le quotient de la division euclidienne de* a *par* b
$\lceil x \rceil$	*Fonction renvoyant la partie entière par excès de* x
$\lfloor x \rfloor$	*Fonction renvoyant la partie entière par défaut de* x
$(a)_{[b]}$	*Fonction modulo renvoyant le reste de la division euclidienne de* a *par* b
$o(x)$	*Fonction de prépondérance indiquant le comportement d'une fonction par rapport à la variable* x *(Notation de Landau)*
$\Pr\{A\}$	*Probabilité de l'évènement* A
$\operatorname{sgn}(a)$	*Fonction renvoyant le signe de* a
$\mathbb{R}$	*Ensemble des réels*
$[\![a; b]\!]$	*Ensemble des entiers exactement compris entre* a *et* b
$[\![a; b[\![$	*Ensemble des entiers exactement compris entre* a *et* $b-1$
$\arg\max(A)$	*Argument qui maximise l'ensemble* A
$a \oplus b$	*Opération XOR effectuée entre les éléments binaires* a *et* b
$card(A)$	*Fonction renvoyant le cardinal de l'ensemble* A

B.2 Notations génériques de codage

K	*Nombre d'éléments binaires contenus dans un message source*
N	*Nombre d'éléments binaires contenus dans un message codé*
N_m	*Nombre d'éléments contenant un message de modulation numérique*
M	*Nombre d'éléments de parité ajoutés lors de l'étape de codage*
R	*Rendement du code*
$\boldsymbol{d}_1^K$	*Message source*
$\boldsymbol{c}_1^N$	*Message codé ou mot de code transmis*
$\boldsymbol{x}_1^{N_m}$	*Message de modulation numérique transmis sur le canal*

$\boldsymbol{y}_1^{N_m}$	*Message de modulation numérique reçu du canal*
$\hat{\boldsymbol{c}}_1^N$	*Décision dure du message codé en sortie du démodulateur*
$\hat{\boldsymbol{d}}_1^K$	*Décision dure du message source en sortie de décodage*
$\mathcal{C}$	*Espace des mots de code*
$\mathcal{A}$	*Alphabet de codage*
n_m	*Nombre de bits codés par symbole de modulation numérique*
d_{min}	*Distance minimale d'un code*
E_b	*Énergie nécessaire à l'émission d'un bit d'information*
E_s	*Énergie nécessaire à l'émission d'un symbole numérique*
N_0	*Densité du spectre de bruit*

B.3 Notations génériques de décodage

$\mathrm{L}^a(c_n)$	*Log-Rapport de Vraisemblance a posteriori de l'élément c_k du mot de code*
$\mathrm{L}^c(c_n)$	*Log-Rapport de Vraisemblance intrinsèque du canal relatif à la variable c_n*
$\mathrm{L}^e(s_k)$	*Log-Rapport de Vraisemblance extrinsèque relatif à l'information systématique s_k*
$\alpha_k(s)$	*Probabilité que la machine de Markov soit à l'état s à l'instant k connaissant l'ensemble des informations du treillis aux états précédents*
$\beta_k(s)$	*Probabilité que la machine de Markov soit à l'état s à l'instant k connaissant l'ensemble des informations du treillis aux états suivants*
$\gamma_k(s, s')$	*Probabilité de la transition entre l'état s et l'état s' à l'instant k sur une machine de Markov connaissant les éléments systématiques et redondantes de l'instant k*
$\lambda_k(s, s')$	*Probabilité de la transition entre l'état s et l'état s' à l'instant k sur une machine de Markov connaissant l'ensemble des probabilités d'états*

B.4 Notations des codes LDPC et turbocodes

H	*Matrice de parité*
$\mathcal{N}_m$	*Ensemble des indices des variables associées à l'équation e_m d'une matrice de parité*
$\mathcal{M}_n$	*Ensemble des indices des équations de parité impliquant la variable c_n*
$d_c^{\mathbf{m}}$	*Degré de parité associé à l'équation e_m d'une matrice de parité*

$d_v^{\mathbf{n}}$	*Degré de variable associé à la variable c_n d'un mot de code*
$\mathbf{d}^H$	*Nombre de nœuds de connexion de la matrice de parité*
$\mathcal{C}^{\mathbf{m}}$	*Sous-groupe d'équations de parité associé à une matrice de parité H d'un code QC-LDPC*
I_j	*Matrice identité permutée de j rang vers la droite*
z	*Facteur d'expansion des matrices QC-LDPC*
ν	*Nombre de mémoire associé à un code convolutif*
$\boldsymbol{H}(z)$	*Ensemble des transformées en z des fonctions de convolution associées à un code convolutif*
m	*Nombre d'éléments systématique en entré d'un code convolutif*
$s_k = \left(s_k^1, ..., p_k^m\right)$	*Éléments systématiques associés à la section k du treillis*
$p_k = \left(p_k^1, ..., p_k^n\right)$	*Éléments de redondance associés à la section k du treillis*
S_k	*État d'une machine de Markov à l'instant k*
M_i	*Valeur de l'élément de mémoire i associé à un code convolutif*
T_l	*Taille du treillis associé à un code convolutif*
$\Pi(k)$	*Fonction d'entrelacement*

B.5 Notations sur l'évaluation matérielle

$\chi_\star^N$	*Complexité de calcul attribuée à l'opérateur $\star$ pour N opérandes*
δ_{pip}	*Délai nécessaire au vidage des pipelines du processeur SISO*
Q_b	*Nombre de bits de quantification*
$\delta_\star^N$	*Délai de calcul attribué à l'opérateur $\star$ pour N opérandes*
f_{clk}	*Fréquence de fonctionnement de l'horloge*
T_{clk}	*Période du cycle d'horloge*

B.6 Notations de l'architecture

f_Λ	*Facteur d'échelonnage de l'information* a posteriori
f_Z	*Facteur d'échelonnage de l'information extrinsèque*
$\mathbf{X}_k$	*Vecteur d'informations intrinsèques définit en entrée ducœur*
$\mathbf{X}_k^{\mathbf{a}}$	*Composante du vecteur d'informations intrinsèques*
$\mathbf{Z}_k^{in}$	*Vecteur d'informations extrinsèques définit en entrée ducœur*
$\mathbf{Z}_k^{in,\mathbf{a}}$	*Composante du vecteur d'informations extrinsèques*
$\boldsymbol{\Lambda}_k$	*Vecteur d'information des décisions sur les variables associées à la section k*

$\mathbf{Z}_k^{out}$	*Vecteur d'information extrinsèque obtenue du décodage de la section k*
$\texttt{MBR}_k^{s,p}(i, p_i)$	*Métrique de branche associée à l'information systématique i et à la redondance p_i*
$\texttt{MBR}_k^{s}(s_k)$	*Coût de la métrique de branche associée à l'information systématique s_k*
$\texttt{MBR}_k^{p}(p_k)$	*Coût de la métrique de branche associée à l'information redondante p_k*
$\texttt{MAC}_k(s)$	*Métrique cumulée de l'état s Aller ou Retour associé à la section k*
$\texttt{MAC}_{act}(s')$	*Métrique cumulée suivante*
$\texttt{MAC}_{pre}(s')$	*Métrique cumulée précédemment calculée*
$\texttt{MAC}_k^{\texttt{FIFO}}(s)$	*Métrique cumulée de l'instant k stockée dans une FIFO*
$\texttt{DEC}_k^{i}(s)$	*Log-probabilité de l'élément i relativement à l'état s en sortie de décodage de treillis*
$\texttt{MEM}_{\mathbf{a}}$	*Banc de mémoire a*

B.7 Notations de repères de treillis

k	*Indice de treillis d'un code convolutif*
κ	*Indice de treillis sur une représentation de treillis quelconque*
$\mathbf{b}$	*Facteur de contrainte multiple*
b	*Indice horizontal associé à une représentation treillis d'un code QC-LDPC*
π_m	*Fonction de correspondance entre l'indice d'une variable c_n et l'indice de treillis k pour l'équation e_m*
$\Pi(\kappa, b)$	*Fonction d'entrelacement liée au code QC-LDPC pour la section équivalente de treillis κ et l'indice de variable b*
T_{It}	*Longueur du treillis pour une itération de décodage*
$T_l^{\mathbf{m}}$	*Longueur du treillis d'ensemble de contraintes indépendantes du sous-code $\mathcal{C}^{\mathbf{m}}$*
w	*Longueur de la fenêtre de décodage*
T_e	*Longueur d'entraînement du décodeur*

Annexe C

Algorithme Somme-Produit pour équation de parité

démonstration

Montrons par récurrence qu'une équation de parité (C.1) implique la relation (C.2)

$$e_n = c_1 \oplus c_2 \oplus ... \oplus c_n \tag{C.1}$$

$$\tanh\left(\frac{1}{2} \times \frac{\Pr\{e_n = 1|c_1, ...c_n\}}{\Pr\{e_n = 0|c_1, ...c_n\}}\right) = \sum_{i=1}^{n} \tanh\left(\frac{1}{2} \times \frac{\Pr\{c_i = 1\}}{\Pr\{c_i = 0\}}\right) \tag{C.2}$$

• Démontrons cette relation pour $n = 2$:
Dans ce cas, l'équation (C.1) se traduit par (C.3).

$$e_2 = c_1 \oplus c_2 \tag{C.3}$$

On définit q_2 selon la relation (C.4) et p_i par (C.5).

$$q_2 = \Pr\{e_2 = 1|c_1, c_2\} \tag{C.4}$$

$$p_i = \Pr\{c_i = 1\} \tag{C.5}$$

Dans ce cas, on pose A_2 suivant la relation (C.6)

$$A_2 = \tanh\left(\frac{1}{2} \times \log\left(\frac{\Pr\{e_n = 0|c_1, c_2\}}{\Pr\{e_n = 1|c_1, c_2\}}\right)\right) \tag{C.6}$$

Développons A_2 en fonction de q_2 :

$$\begin{aligned}
A_2 &= \tanh\left(\tfrac{1}{2} \times \log\left(\tfrac{\Pr\{e_n=0|c_1,c_2\}}{\Pr\{e_n=1|c_1,c_2\}}\right)\right) \\
A_2 &= \tanh\left(\tfrac{1}{2} \times \log\left(\tfrac{1-q_2}{q_2}\right)\right) \\
A_2 &= \frac{e^{\log\left(\frac{1-q_2}{q_2}\right)}-1}{e^{\log\left(\frac{1-q_2}{q_2}\right)}+1} \\
A_2 &= \left(\tfrac{1-q_2}{q_2} - \tfrac{q_2}{q_2}\right)/\left(\tfrac{1-q_2}{q_2} + \tfrac{q_2}{q_2}\right) \\
A_2 &= 1 - 2.q_2
\end{aligned} \tag{C.7}$$

Or,

$$\begin{array}{rcc} q_2 & = & \Pr\{e_2 = 1|c_1, c_2\} \\ q_2 & = & \Pr\{(c_1 = 1, c_2 = 0) \bigcup (c_1 = 0, c_2 = 1)\} \\ q_2 & = & p_1.(1-p_2) + p_2.(1-p_1) \end{array} \tag{C.8}$$

En remplaçant q_2 dans l'expression (C.7) on obtient :

$$\begin{array}{rcc} A_2 & = & 1 - 2.p_1.(1-p_2) - 2.p_2.(1-p_1) \\ A_2 & = & 1 + 2.p_2 + 2.p_1 - 4.p_1.p_2 \\ A_2 & = & (1-2.p_1) \times (1-2.p_2) \end{array} \tag{C.9}$$

En appliquant le raisonnement détaillé dans la relation (C.7) à la relation (C.9), il advient :

$$\tanh\left(\frac{1}{2} \times \frac{\Pr\{e_2 = 1|c_1, c_2\}}{\Pr\{e_2 = 0|c_1, c_2\}}\right) = \tanh\left(\frac{1}{2} \times \frac{\Pr\{c_1 = 1\}}{\Pr\{c_1 = 0\}}\right) \times \tanh\left(\frac{1}{2} \times \frac{\Pr\{c_2 = 1\}}{\Pr\{c_2 = 0\}}\right) \tag{C.10}$$

La proposition est donc vérifiée pour $n = 2$.

- On suppose la proposition vérifiée à l'ordre n. Démontrons le pour l'ordre $n+1$:

Dans ce cas (C.1) équivaut aux relations (C.11-C.12).

$$e_{n+1} = c_1 \oplus c_2 \oplus ... \oplus c_n \oplus c_{n+1} \tag{C.11}$$

$$e_{n+1} = e_n \oplus c_{n+1} \tag{C.12}$$

Par supposition, on sait que $A_n = \prod_{i=1}^{n} (1 - 2.p_i)$. Par un raisonnement similaire à celui développé dans (C.7), on obtient alors

$$\begin{array}{rcc} A_{n+1} & = & \tanh\left(\frac{1}{2} \times \log\left(\frac{\Pr\{e_{n+1}=0|c_1,...c_n,c_{n+1}\}}{\Pr\{e_{n+1}=1|c_1,...c_n,c_{n+1}\}}\right)\right) \\ A_{n+1} & = & \tanh\left(\frac{1}{2} \times \log\left(\frac{\Pr(e_{n+1}=0|e_n,c_{n+1})}{\Pr\{e_{n+1}=1|e_n c_{n+1}\}}\right)\right) \\ & ... & \\ A_{n+1} & = & (1-2.p_{n+1}) \times (1-2.q_n) \end{array} \tag{C.13}$$

Or,

$$A_n = (1-2.q_n)$$

Nous retrouvons bien

$$A_{n+1} = (1-2.p_{n+1}) \times A_n$$

Donc la proposition est vérifiée pour $n+1$.

- La proposition 1.4.1 a été démontrée par récurrence.

Bibliographie

[1] C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27 :379–423 and 623–656, jul 1948. (Cité en pages 4 et 14.)

[2] R. W. Hamming. Error-detecting and correcting codes. *Bell Systems Technical Journal (BSTJ)*, page 147â160, apr 1950. (Cité en page 4.)

[3] M. J. E. Golay. Notes on Digital Coding. *Proc. IRE*, apr 1949. (Cité en page 4.)

[4] R. C. Bose and D. K. Ray Chaudhuri. On a class or error-correcting binary group codes. *Information and Control*, apr 1960. (Cité en page 4.)

[5] IEEE Standards Association. *IEEE Std 802.15.1-2005 - Part 15.1 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)*, jun 2011. (Cité en page 4.)

[6] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial & Applied Mathematics*, 8(2) :300–304, 1960. (Cité en page 4.)

[7] European Telecommunications Standards Institute (ETSI) - EN 300 429 V1.2.1. *Digital Video Broadcasting (DVB) ; Framing structure, channel coding and modulation for cable systems*, avr 1998. (Cité en page 4.)

[8] European Telecommunications Standards Institute (ETSI) - EN 300 744 V1.6.1. *Digital Video Broadcasting (DVB) ; Framing structure, channel coding and modulation for digital terrestrial television*, jan 2009. (Cité en page 4.)

[9] European Telecommunications Standards Institute (ETSI) - EN 300 421 V1.1.2. *Digital Video Broadcasting (DVB) ; Framing structure, channel coding and modulation for 11/12 GHz satellite services*, aug 1997. (Cité en page 4.)

[10] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding : Turbo-codes. In *IEEE International Conference on Communications. ICC 93.*, pages 1064 –1070, may 1993. (Cité en pages 4, 22, 30 et 32.)

[11] R. G. Gallager. *Low Density Parity-Check Codes* . PhD thesis, MIT Press, Cambridge, 1963. (Cité en pages 4, 14, 24, 27, 29, 54 et 60.)

[12] European Telecommunications Standards Institute (ETSI) - EN 302 307 V1.2.1. *Digital Video Broadcasting (DVB) ; Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2)*, aug 2009. (Cité en page 4.)

[13] M.P.C. Fossorier. Quasicyclic low-density parity-check codes from circulant permutation matrices. *IEEE Transactions on Information Theory*, 50(8) :1788 –1793, aug 2004. (Cité en pages 5, 14 et 17.)

[14] IEEE Computer Society. *IEEE Standard for Information technology – Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 5 : Enhancementss for Higher Throughput*, oct 2009. (Cité en pages 5, 7, 122 et 127.)

[15] IEEE Computer Society. *IEEE Draft Standard for Information technology – Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 5 : Enhancementss for Higher Throughput for Operation in Bands below 6 GHz*, jan 2013. (Cité en pages 5, 7 et 127.)

[16] IEEE Computer Society. *IEEE P802.11ad : IEEE Draft Standard for Information technology – Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 3 : Enhancements for Very High Throughput in the 60 GHz Band*, jul 2012. (Cité en pages 5, 7 et 127.)

[17] IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks – Part 16 : Air Interface for Broadband Wireless Access Systems*, may 2009. (Cité en pages 5, 6, 90, 127 et 128.)

[18] IEEE Computer Society. *IEEE Standard for Information technology – Part 15.3 : Wireless Medium Access Control (MAC) and Physical (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*, oct 2009. (Cité en pages 5 et 127.)

[19] Telecommunication standardization sector of ITU. *G9960 - Series G : Transmission Systems and Media, Digital Systems and Networks - Unified high-speed wire-line based homenetworking transceivers - System architectureand physical layer specification*, jui 2010. (Cité en pages 5, 7 et 127.)

[20] Telemetry Channel Coding. Consultative committee for space data systems, october 2002, recommendation for space data system standards. Technical report, CCSDS 101.0-B-6, Blue Book, October 2002. (Cité en pages 6, 90 et 128.)

[21] 3rd Generation Partnership Project (3GPP) - TS 25.212 - V9.4.0. *Group Radio Access Network ; Multiplexing and channel coding (FDD) (Release 9)*, dec 2010. (Cité en pages 6, 7, 90 et 128.)

[22] 3rd Generation Partnership Project (3GPP) - TS 36.212 - V10.0.0. *Group Radio Access Network ; Evolved Universal Terrestrial Radio Access (E-UTRA) ; Multiplexing and channel coding (Release 10)*, dec 2010. (Cité en pages 6, 7, 90, 122 et 128.)

[23] 3rd generation partnership Project 2. *Physical Layer for cdma2000 Extended Cell High Rate Packet Data Air Interface Specification*, oct 2010. (Cité en pages 6 et 90.)

[24] European Telecommunications Standards Institute (ETSI) - EN 301 790 V1.5.1. *Digital Video Broadcasting (DVB) ; Interaction channel for satellite distribution systems*, may 2009. (Cité en pages 6, 90 et 128.)

[25] European Telecommunications Standards Institute (ETSI) - EN 301 958 V1.1.1. *Digital Video Broadcasting (DVB) ; Interaction channel for Digital Terrestrial Television (RCT) incorporating Multiple Access OFDM*, mar 2002. (Cité en pages 6, 90 et 128.)

[26] European Telecommunications Standards Institute (ETSI) - EN 302 583 V1.1.2. *Digital Video Broadcasting (DVB) ; Framing Structure, channel coding and modulation for Satellite Services to Handheld devices (SH) below 3 GHz*, sep 2010. (Cité en pages 6 et 128.)

[27] European Telecommunications Standards Institute (ETSI) - EN 101 3 545-3 V1.1.1. *Digital Video Broadcasting (DVB) ; Second Generation DVB Interactive Satellite System ; Part 2 : Lower Layers for Satellite standard*, mar 2011. (Cité en pages 6, 90 et 128.)

[28] HomePlug AV Specification. *Draft HomePlug AV Specification, Version 1.1* , May 2007. (Cité en pages 6, 7, 91 et 128.)

[29] HomePlug AV Specification. *HomePlug AV Draft Spec Version 2.m* , dec 2011. (Cité en pages 6 et 7.)

[30] IEEE Computer Society. *IEEE Standard 802.3AN-2006 : Specific requirements Part 3 : Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment 1 : Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T*, sep 2006. (Cité en page 7.)

[31] IEEE Std 1901-2010. *IEEE Standard for Broadband over Power Line Networks : Medium Access Control and Physical Layer Specifications*, sep 2010. (Cité en page 7.)

[32] IEEE Standard association. *Overview of P1905.1, white paper*, dec 2011. (Cité en page 7.)

[33] D.J.C MacKay and R.M Neal. Good Codes based on Very Sparse Matrices. In *5th IMA Conference on Cryprography and Coding*, 1995. (Cité en page 14.)

[34] D.J.C. MacKay. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2) :399 – 431, mar 1999. (Cité en page 14.)

[35] N. Wiberg, H.-A. Loeliger, and R. Kotter. Codes and iterative decoding on general graphs. In *IEEE International Symposium on Information Theory, 1995.*, page 468, sep 1995. (Cité en page 14.)

[36] T.J. Richardson and R.L. Urbanke. Efficient encoding of low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2) :638 –656, feb 2001. (Cité en pages 14, 15 et 16.)

[37] S. Ten Brink. Convergence of iterative decoding. *Electronics Letters*, 35(10) :806 –808, may 1999. (Cité en page 14.)

[38] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5) :533 – 547, sep 1981. (Cité en pages 15 et 16.)

[39] P. Elias. Coding for noisy channels. *IRE Conv. Rec*, 3(4) :37–46, 1955. (Cité en page 18.)

[40] H. Ma and J. Wolf. On Tail Biting Convolutional Codes. *IEEE Transactions on Communications*, 34(2) :104 – 111, feb 1986. (Cité en page 21.)

[41] C. Berrou, C. Douillard, and M. Jézéquel. Multiple parallel concatenation of circular recursive systematic convolutional (CRSC) codes. In *Annales des télécommunications*, volume 54, pages 166–172. Springer, 1999. (Cité en page 21.)

[42] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding : turbo-codes. *IEEE Transactions on Communications*, 44(10) :1261–1271, 1996. (Cité en page 22.)

[43] O.Y. Takeshita and Costello D.J. New deterministic interleaver designs for turbo codes. *IEEE Transactions on Information Theory*, 46(6) :1988–2006, 2000. (Cité en page 22.)

[44] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel. Designing good permutations for turbo codes : towards a single model. In *ICC*, pages 341–345, 2004. (Cité en page 23.)

[45] O.Y. Takeshita. A new construction for LDPC codes using permutation polynomials over integer rings. *arXiv preprint cs/0506091*, 2005. (Cité en page 23.)

[46] S. Crozier and P. Guinand. Distance upper bounds and true minimum distance results for turbo-codes designed with DRP interleavers. In *Annales des télécommunications*, volume 60, pages 10–28. Springer, 2005. (Cité en pages 23 et 48.)

[47] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2) :260 – 269, apr 1967. (Cité en pages 24 et 30.)

[48] F. Guilloud. *Generic Architecture for LDPC Codes Decoding* . PhD thesis, Telecom Paris, 2004. (Cité en pages 26 et 27.)

[49] M.P.C. Fossorier, M. Mihaljevic, and H. Imai. Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation . *IEEE Transactions on Communications*, pages 673 – 680, may 1999. (Cité en pages 26 et 27.)

[50] J. Chen and M.P.C. Fossorier. Density evolution for two improved BP-Based decoding algorithms of LDPC codes. *IEEE Communications Letters*, 6 :208 –210, may 2002. (Cité en pages 26 et 27.)

[51] Y. Mao and A.H. Banihashemi. Decoding low-density parity-check codes with probabilistic schedule. In *2001 IEEE Pacific Rim Conference on Communications, Computers and signal Processing, 2001. PACRIM*, volume 1, pages 119 –123 vol.1, 2001. (Cité en page 29.)

[52] H. Kfir and I. Kanter. Parallel versus sequential updating for belief propagation decoding. *Physica A*, 2003. (Cité en pages 29 et 60.)

[53] Y.M. Chang, A.I. Vila Casado, M.-C.F. Chang, and R.D. Wesel. Lower-Complexity Layered Belief-Propagation Decoding of LDPC Codes. In *IEEE International Conference on Communications, 2008. ICC '08*, pages 1155 – 1160, may 2008. (Cité en page 29.)

[54] J. Hagenauer and P. Hoeher. A Viterbi algorithm with soft-decision outputs and its applications . In *Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM '89., IEEE*, pages 1680 –1686 vol.3, nov 1989. (Cité en page 30.)

[55] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, 20(2) :284 – 287, mar 1974. (Cité en page 30.)

[56] P.J. Black and T.H.-Y. Meng. "A 140 Mb/s 32-state radix-4 Viterbi decoder". In *Solid-State Circuits Conference, 1992. Digest of Technical Papers. 39th ISSCC, 1992 IEEE International*, pages 70 –71, 247, feb 1992. (Cité en page 34.)

[57] C. Roth, P. Meinerzhagen, C. Studer, and A. Burg. A 15.8 pJ/bit/iter quasi-cyclic LDPC decoder for IEEE 802.11 n in 90 nm CMOS. In *Solid State Circuits Conference (A-SSCC), 2010 IEEE Asian*, pages 1–4. IEEE, 2010. (Cité en pages 35, 37 et 154.)

[58] T. Brack, M. Alles, F. Kienle, and N. Wehn. A Synthesizable IP Core for WIMAX 802.16E LDPC Code Decoding. *2006 IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1 –5, sep 2006. (Cité en pages 35 et 37.)

[59] G. Masera, F. Quaglio, and F. Vacca. Implementation of a Flexible LDPC Decoder. *IEEE Transactions on Circuits and Systems II : Express Briefs*, 54(6) :542 –546, jun 2007. (Cité en pages 36 et 37.)

[60] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol. A 24Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless. *Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International*, 1 :150 – 484, feb 2003. (Cité en pages 36 et 37.)

[61] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang. A 58mW 1.2mm^2 HSDPA Turbo Decoder ASIC in 0.13μm CMOS. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 264 – 612, feb 2008. (Cité en page 36.)

[62] Y. Sun and J.R. Cavallaro. Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder. *INTEGRATION, the VLSI journal*, 44(4) :305–315, 2011. (Cité en pages 36 et 37.)

[63] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn. A 2.15 GBit/s turbo code decoder for LTE advanced base station applications. In *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 21–25. IEEE, 2012. (Cité en pages 36 et 37.)

[64] R. Doe. Homeplug AV FEC product, jun 2013. (Cité en pages 36 et 37.)

[65] J.H. Kim and I.C. Park. A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE. In *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE*, pages 487 –490, sep 2009. (Cité en pages 36 et 37.)

[66] M. Alles, T. Vogt, and N. Wehn. FlexiChaP : A reconfigurable ASIP for convolutional, turbo, and LDPC code decoding. In *2008 5th International Symposium on Turbo Codes and Related Topics* , pages 84 –89, sep 2008. (Cité en pages 36 et 38.)

[67] F. Naessens, V. Derudder, H. Cappelle, L. Hollevoet, P. Raghavan, M. Desmet, A.M. AbdelHamid, I. Vos, L. Folens, S. O'Loughlin, S. Singirikonda, S. Dupont, J.-W. Weijers, A. Dejonghe, and L. Van der Perre. A 10.37 mm2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE. In *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*, pages 213 –214, jun 2010. (Cité en pages 36 et 38.)

[68] Y. Sun and J.R. Cavallaro. Unified decoder architecture for LDPC/turbo codes. In *IEEE Workshop on Signal Processing Systems, 2008. SiPS 2008*, volume 2, pages 13 –18, oct 2008. (Cité en pages 38, 47, 154 et 155.)

[69] G. Gentile, M. Rovini, and L. Fanucci. A multi-standard flexible turbo/LDPC decoder via ASIC design. In *2010 6th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 294 –298, sep 2010. (Cité en pages 38, 154 et 155.)

[70] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang. Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE. *J. Solid-State Circuits*, 46(1) :8–17, 2011. (Cité en pages 38, 69, 153 et 154.)

[71] Xilinx. *Virtex-6 Family Overview, DS150 (v2.4)* , jan 2012. (Cité en page 42.)

[72] D. P. Mehta. *Handbook of data structures and applications*. CRC Press, 2004. (Cité en page 43.)

[73] W.F. Sheppard. On the calculation of the most probable values of frequency-constants, for data arranged according to equidistant division of a scale. *Proceedings of the London Mathematical Society*, 1(1) :353–380, 1897. (Cité en page 45.)

[74] E. Abaya and G.L. Wise. Some notes on optimal quantization. In *ICC'81 ; International Conference on Communications, Volume 2*, volume 2, page 30, 1981. (Cité en page 45.)

[75] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6) :2325–2383, 1998. (Cité en page 45.)

[76] D.J. Lilja and S.S. Sapatnekar. *Designing Digital Computer Systems with Verilog*. Cambridge University Press, 2005. (Cité en page 46.)

[77] M. Senthilvelan, M. Yu, D. Iancu, M. Sima, and M. Schulte. CORDIC instructions for LDPC decoding on SDR platforms. *Analog Integrated Circuits and Signal Processing*, 69(2-3) :191–206, 2011. (Cité en page 47.)

[78] J. E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Transactions on Electronic Computers*, 8(3) :330 –334, sep 1959. (Cité en page 47.)

[79] G. Masera, F. Quaglio, and F. Vacca. Finite precision implementation of LDPC decoders. *Communications, IEE Proceedings*, 152(6) :1098–1102, 2005. (Cité en page 47.)

[80] X.Y. Hu, M.P.C. Fossorier, and E. Eleftheriou. On the computation of the minimum distance of low-density parity-check codes. In *IEEE International Conference on Communications, 2004*, volume 2, pages 767–771. IEEE, 2004. (Cité en page 48.)

[81] X.Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia. Efficient implementations of the sum-product algorithm for decoding LDPC codes. In *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE*, volume 2, pages 1036–1036E. IEEE, 2001. (Cité en page 53.)

[82] C. Marchand, L. Conde-Canencia, and E. Boutillon. Architecture and finite precision optimization for layered LDPC decoders. *Journal of Signal Processing Systems*, 65(2) :185–197, 2011. (Cité en page 54.)

[83] D.E. Hocevar. A reduced complexity decoder architecture via layered decoding of LDPC codes. In *IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004*, pages 107 – 112, oct 2004. (Cité en page 60.)

[84] S. Bitterlich and H. Meyr. Efficient scalable architectures for Viterbi decoders. In *International Conference on Application-Specific Array Processors, 1993. Proceedings.*, pages 89–100, 1993. (Cité en page 69.)

[85] M.P.C. Fossorier and Shu L. Differential trellis decoding of convolutional codes. *IEEE Transactions on Information Theory*, 46(3) :1046–1053, 2000. (Cité en pages 69 et 122.)

[86] E. Yeo, S.A. Augsburger, W.R. Davis, and B. Nikolic. A 500-Mb/s soft-output Viterbi decoder. *IEEE Journal of Solid-State Circuits*, 38(7) :1234–1241, 2003. (Cité en page 69.)

[87] S.J. Lee, N.R. Shanbhag, and A.C. Singer. A 285-MHz pipelined MAP decoder in 0.18μm CMOS. *IEEE Journal of Solid-State Circuits*, 40(8) :1718 – 1725, aug 2005. (Cité en page 69.)

[88] M. Martina, M. Nicola, and G. Masera. A Flexible UMTS-WiMax Turbo Decoder Architecture. *IEEE Transactions on Circuits and Systems II : Express Briefs*, 55(4) :369 –373, apr 2008. (Cité en page 69.)

[89] Yuping Zhang and K.K. Parhi. High-Throughput Radix-4 logMAP Turbo Decoder Architecture. In *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, pages 1711 –1715, nov 2006. (Cité en page 69.)

[90] M. May, T. Ilnseher, N. Wehn, and W. Raab. A 150Mbit/s 3GPP LTE Turbo code decoder. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 1420 –1425, mar 2010. (Cité en page 69.)

[91] G. Masera, G. Piccinini, M.R. Roch, and M. Zamboni. VLSI architectures for Turbo codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(3) :369 –379, sep 1999. (Cité en pages 69 et 109.)

[92] T.S.V. Gautham, A. Thangaraj, and D. Jalihal. Common architecture for decoding turbo and LDPC codes. *National Conference on Communications (NCC), 2010*, pages 1 – 5, jan 2010. (Cité en page 77.)

[93] M.M. Mansour and N.R. Shanbhag. Turbo decoder architectures for low-density parity-check codes. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 2, pages 1383 – 1388, nov 2002. (Cité en page 77.)

[94] M. Chiu. Low-Density Parity-Check Codes with 2-State Trellis Decoding. *IEEE Transactions on Communications*, 57 :12 –16, jan 2009. (Cité en page 77.)

[95] O. Muller, A. Baghdadi, and M. Jezequel. On the Parallelism of Convolutional Turbo Decoding and Interleaving Interference. In *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1–5, 2006. (Cité en pages 99 et 138.)

[96] S. Haddad, G. Sanchez, D. Oscar, A. Baghdadi, and M. Jezequel. Complexity reduction of shuffled parallel iterative demodulation with turbo decoding. In *ICT 2012 : 19th International Conference on Telecommunications*, 2012. (Cité en page 99.)

[97] P.J. Black and T.H.-Y. Meng. A 1-Gb/s, four-state, sliding block Viterbi decoder. *IEEE Journal of Solid-State Circuits*, 32(6) :797–805, 1997. (Cité en page 101.)

[98] D. Gnaedig, E. Boutillon, J. Tousch, and M. Jezequel. Towards an optimal parallel decoding of turbo codes. In *4th International Symposium on Turbo Codes Related Topics ; 6th International ITG-Conference on Source and Channel Coding (TURBOCODING), 2006*, pages 1–6, 2006. (Cité en page 101.)

[99] O. Muller, A. Baghdadi, and M. Jezequel. ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding. *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, 1 :1 – 6, mar 2006. (Cité en page 101.)

[100] O. Muller. *Architectures multiprocesseurs monopuces génériques pour turbo-communications haut-débit* . PhD thesis, TELECOM Bretagne, 2007. (Cité en page 103.)

[101] D. Gnaedig. *Optimisation des architectures de décodage des turbo-codes* . PhD thesis, TELECOM Bretagne, 2005. (Cité en page 103.)

[102] A. Tarable, S. Benedetto, and G. Montorsi. Mapping interleaving laws to parallel turbo and LDPC decoder architectures. *Information Theory, IEEE Transactions on*, 50(9) :2002 – 2009, sep 2004. (Cité en page 103.)

[103] O.Y. Takeshita. On maximum contention-free interleavers and permutation polynomials over integer rings. *IEEE Transactions on Information Theory*, 52(3) :1249 –1253, mar 2006. (Cité en page 103.)

[104] J. Zhang and M.P.C. Fossorier. Shuffled iterative decoding. *IEEE Transactions on Communications*, 53(2) :209–213, 2005. (Cité en page 106.)

[105] J. Dion, M.-H. Hamon, P. Penard, M. Arzel, and M. Jezequel. Adapted scheduling of QC-LDPC decoding for multistandard receivers. In *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 111–115, 2012. (Cité en pages 110 et 115.)

[106] J. Dion, M.-H. Hamon, P. Penard, M. Jezequel, and M. Arzel. Multi-standard Trellis-based FEC Decoder. In *DASIP 2012 : Conference on Design and Architectures for Signal and Image Processing*, 2012. (Cité en page 114.)

[107] I. Tanyanon and S. Choomchuay. A hardware design of MS/MMS-based LDPC decoder. In *IEEE International Conference onElectron Devices and Solid State Circuit (EDSSC)*, pages 1–4, 2012. (Cité en page 148.)

[108] T. Adiono and Marvin. Radix-4 Max-log-MAP parallel turbo decoder architecture with a new cache memory data flow for LTE. In *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on*, pages 792–797, 2012. (Cité en page 148.)

[109] Y. Huang, C. Chen, C. Zhou, Chen Y., and Zeng X. A common flexible architecture for Turbo/LDPC codes. In *SoC Design Conference (ISOCC), 2011 International*, pages 54–57, nov 2011. (Cité en pages 154 et 155.)

[110] C.-C. Wong, M.-W. Lai, C.-C. Lin, H.-C. Chang, and C.-Y. Lee. Turbo decoder using contention-free interleaver and parallel architecture. *IEEE Journal of Solid-State Circuits*, 45(2) :422–432, 2010. (Cité en pages 154 et 155.)

[111] T.-C. Kuo and A.N. Willson. A flexible decoder IC for WiMAX QC-LDPC codes. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, pages 527–530, 2008. (Cité en pages 154 et 155.)